Modular Design Architecture

Accelerating Product Design Processes through Modularity

Christian Anttonen

Modular Design Architecture

Accelerating Product Design Processes through Modularity

Christian Anttonen

Aalto University
School of Arts, Design
and Architecture

AUTHOR Christian Anttonen **TITLE OF THESIS** Modular Design Architecture: Accelerating Product Design Processes through Modularity **DEPARTMENT** Media Department, Media Lab **DEGREE PROGRAMME** New Media Design and Production **NUMBER OF PAGES** 107 + 10 **LANGUAGE** English **SUPERVISOR** Markku Reunanen **ADVISOR** Ville Eloranta **YEAR** Spring 2020

Abstract

Finding the line between time investment in pre-designing product architecture instead of creating it only when needed is complex; if the foundation is rashly defined, it results in a high chance of failure. On the other hand, strict planning means time away from production, and thus, the postponement of a product launch. Moreover, a restricted foundation can make responding to future changes challenging.

Modular design architecture, together with modular product architecture, enables companies to design, iterate, and push their products faster to the markets. Since the building blocks for agile product design and development are readily available, approximately all robust products are modular nowadays.

This thesis addresses and inspects how modular design architecture can accelerate product design and development processes when creating new web-based products. The researched literature presents insights and theoretical frameworks regarding modularity, software development, and product design. It additionally addresses teamwork and best practices found in product development.

The research was conducted based on the grounded theory methodology. Its primary data collection method was benchmarking, and it applied the grounded theory data coding methods to scrutinize and categorize the acquired data. This thesis coins a novel framework based on the research findings to address interrelationships amidst various definitions, elements, and categories found within modular design architectures.

KEYWORDS: Modularity, Product Design, Product Development, Design Systems, Design Patterns, UI Frameworks, User Interfaces

Aalto University
School of Arts, Design
and Architecture

TEKIJÄ Christian Anttonen **TYÖN NIMI** Modular Design Architecture: Accelerating Product Design Processes through Modularity **LAITOS** Median laitos, Media Lab **KOULUTUSOHJELMA** New Media Design and Production **SIVUMÄÄRÄ** 107 + 10 **KIELI** Englanti **VALVOJA** Markku Reunanen **OHJAAJA** Ville Eloranta **VUOSI** Kevät 2020

Tiivistelmä

Täydellisen linjan löytäminen siinä, kuinka paljon aikaa tulisi investoida tuotearkkitehtuurin suunnitteluun etukäteen sen sijaan, että sitä luotaisiin vain tarvittaessa, on hankalaa; jos tuotteen pohjatyö määritetään liian harkitsemattomasti, se luo suuremman riskin epäonnistumiseen. Toisaalta, liian yksityiskohtainen suunnittelu vie aikaa pois tuotannosta, lykäten tuotteen lanseerausta. Tämän lisäksi liian rajoitettu tuotearkkitehtuuri voi tehdä reagoimisen tuleviin muutoksiin haastaviksi.

Modulaarinen suunnitteluarkkitehtuuri yhdessä modulaarisen tuotearkkitehtuurin kanssa mahdollistaa yritysten suunnitella, iteroida ja lanseerata nopeammin tuotteitaan markkinoille. Koska kaikki ketterän tuotesuunnittelun ja kehityksen vaadittavat osa-alueet ovat jo saatavilla, ovat melkein kaikki pitkälle kehitetyt tuotteet modulaarisia.

Tässä työssä tarkastellaan, kuinka modulaarinen suunnitteluarkkitehtuuri voi nopeuttaa tuotesuunnittelu- ja kehitysprosesseja luotaessa uusia verkkopohjaisia tuotteita. Työssä hyödynnetty kirjallisuus tarjoaa näkemyksiä ja teoreettisia viitekehyksiä modulaarisuudesta, ohjelmistokehityksestä ja tuotesuunnittelusta. Edellä mainittujen lisäksi se käsittelee myös ryhmätyötä ja tuotekehityksen parhaita käytäntöjä.

Tutkimus suoritettiin grounded theory -metodologian pohjalta. Sen ensisijainen tiedonkeruumenetelmä oli vertailuanalyysi, ja se hyödynsi grounded theory:n eri koodausmenetelmiä hankitun tiedon tarkastelemiseen ja luokitteluun. Tämä tutkimus luo tutkimustuloksien pohjalta uuden viitekehyksen tarkastelemaan modulaarisen suunnitteluarkkitehtuurin eri määritelmien, elementtien ja luokkien keskinäisiä vuorovaikutussuhteita ja -tasoja.

AVAINSANAT: modulaarisuus, tuotesuunnittelu, tuotekehitys, design systeemit, suunnittelupohjat, käyttöliittymien viitekehykset, käyttöliittymät

Contents

CHAPTER I

Introduction

This introductory chapter provides a brief overview of this thesis. It summarizes its content, presents the ideas behind this study, and describes the structure of this thesis.

Modularity, scalability, and standardization are central themes when it comes to building robust products. Standardization has played a significant role in modularity; scalable and rapid product development would not be as readily achievable without industry-wide standards.[1] Moreover, creating modular design and product architectures for web-based products would not be possible without standards and shared best practices amidst creatives and developers. Modular design and product architectures accelerate product design and development processes while reducing production costs.[2] Modular product architecture enables teams to produce their products by applying the component-by-component construction for different stakeholders to focus on specific parts of the product, fostering seamless and uninterrupted workflows.[3]

However, despite the advantages of modularity being understood, design and its implementation processes are frequently perceived as linear processes. This thesis presents insights, frameworks, and design approaches that are well-acknowledged in the design industry to address this phenomenon. Moreover, the thesis presents a new design framework—a new framework of architectural dependencies within modular design architecture—that is formed based on empirical data acquired during the research process.

The concept of modular design architecture is similar to the concept of design systems. In essence, both focus on establishing a modular and hierarchical structure for a product to ensure that future design decisions, processes, and implementations would be systematic and manageable. They fundamentally differ in their architectural levels: a design system is formed upon a modular design architecture of a product, and the current discussion around design systems concerns

1 Baldwin and Clark 1997, 1–5
2 Gershenson, Prasad and Allamneni 1999, 13
3 Curtis 2010, 9–10

the creation of consistent user interfaces (UIs) for web-based products. In contrast, modular design architecture can be applied to any modular product. However, design systems are nearly limitless entities that range from branding to component-based software engineering, as discussed comprehensively in Section 3.1. Modular design architecture combines themes and topics around design, modularity, and product design, and hence, it provides a well-constructed playground both for design systems and this thesis.

Purpose and Scope

Modular design architecture offers numerous advantages when designing sophisticated web-based products, but architectural issues are rarely thoroughly considered in advance. [4] The purpose of this thesis is to provide insights by guiding designers and other product team members in understanding why it is essential to define modular design architecture at the beginning of a product design process. By combining concepts and frameworks found in academic literature with insights and best practices acquired from the design industry, this thesis aims to form a comprehensive and multidisciplinary approach to the topic.

From the perspective of the product designer, the thesis explores these discussions from the context of designing for the web. Moreover, the thesis provides insights on how to establish a common ground between different stakeholders within a production team to tackle potential issues and bottlenecks when planning modular design architectures for sophisticated web-based products.

The scope of this thesis includes the definition and creation of a modular and scalable design architecture for a web-based product. Closely related topics and themes are covered, such as product and

4 Godbolt 2016, 11

software development, design systems, and business design. However, these topics are only superficially covered to support, clarify, and develop the main content.

Research

This thesis attempts to answer the following research questions: "How can modular design architecture accelerate product design processes when creating sophisticated web-based products?" and "How can modular design architecture be defined when designing new web-based products?"

At the core of its research approach, this thesis applied the grounded theory methodology. The research process consisted of two phases, with each phase a set of sub-phases.

The first phase was a benchmarking process and is the primary data collection method. The applied data was acquired by benchmarking three robust web-based streaming services—Deezer, Tidal, and SoundCloud. The benchmarking process had three main stages. The first stage of the benchmarking process started by defining and examining design patterns that the case companies have applied to their web platforms. The second stage inspected discovered design patterns by dividing them into smaller elements and definitions. The third stage explored modularity and scrutinized the companies' source codes—both Cascading Style Sheets (CSS) and Hypertext Markup Language (HTML) structures—and carried the findings of the first phase of the research process into its second phase.

The second research phase involved data coding and analysis. The data carried from the first research phase was analyzed by applying several coding methods from grounded theory. The second phase additionally had three main stages. The first stage was an open coding process, during which the acquired data was comprehensively

scrutinized. The second stage was an axial coding process that focused on discovering relationships between the categories that were established during the first stage. The third stage was a selective coding process that defined several umbrella terms under which all discovered findings could be classified.

The research addressed how a modular design approach, together with modular design architecture, could accelerate product design and development processes when designing new web-based products. The empirical data acquired throughout the research process formed a new framework that is introduced more broadly as a part of the research findings.

Structure

This thesis consists of eight chapters. A literature review is presented, and the main themes, theories, and frameworks are introduced. The second chapter provides a theoretical approach to the topic and is primarily written from peer-reviewed articles from journals, books, and other publications. In contrast, the third chapter provides a practical approach by introducing the current best practices found in the design industry. The third chapter begins with a brief historical context of the thesis, and its literature is based on expert insights, books, and curated internet sources. The second and third chapters additionally introduce and define the vocabulary around the research topic.

Chapter 4 is an introduction to the research process and completes the previous chapters by introducing applied research methods and the research methodology. The first section of Chapter 4 starts by introducing grounded theory, after which the three coding stages of the data coding process are presented. Chapter 4's second section introduces benchmarking and presents the applied benchmarking methods.

Chapter 5 outlines the research process and puts the introduced methods from the previous chapter into practice. Background information on the research is provided, and the conducted benchmarking process is revealed. After introducing the benchmarking process—the first phase of the research process—the second phase of the research process, the coding phase, is reviewed.

Chapter 6 presents the research findings according to the stages of the research process. Chapter 6 additionally includes research insights that were discovered and formed during the research process.

Chapter 7 discusses the topics and themes found in this thesis, providing personal insights and views regarding the current state of the design industry. It additionally predicts several assessments of future design trends and working methods.

The final chapter, Chapter 8, summarizes the thesis. The research findings are briefly introduced and answers to the research questions are provided. Moreover, the final chapter addresses future research needs and the purpose for them. The thesis ends with an analysis of the study's limitations and validity. The research process, applied methods, and research findings are critically explored.

CHAPTER II

Modularity in Product Design

Modular product architecture enables rapid product development but requires competent management. This chapter provides a theoretical background in product design and development through various frameworks, theories, and insights.

The current academic literature offers various definitions to define the term *modularity*. For instance, David Parnas writes that in software design, one of the characteristics of modularity is the ability to write modules with limited knowledge of the code in other modules. He additionally adds that another characteristic is the capability of reassembling and replacing modules without reassembling the whole system. [5] Karl Ulrich and Steven Eppinger confirm this view: they state that modular architecture allows changes in a module without altering the proper functioning of other modules. They see modules as function-orientated, implying that each module has its own functions that can be implemented into larger systems. [6]

The current literature in (fine) arts offers limited knowledge of modularity and modules. Excluding architecture, the art industry has not significantly applied nor studied modularity. However, notable artists have used modularity in their artwork, such as Mitzi Cunliffe (sculptor, 1918–2006), Erwin Hauer (sculptor, 1926–2017), and Leda Luss Luyken (conceptual artist, 1952), to name a few (see Image 1).

In general, modularity is an attribute that enables products to be built by individual parts that are connectable in numerous different ways. This chapter explores the concept of modularity by dividing the concept into smaller sections and inspecting them individually. The chapter additionally explores the concept of modularity in the context of product design and development. Both product design and product development are introduced in this chapter because product design is inseparable from product development in the context of this thesis. This chapter additionally presents several concepts, models, and processes associated with both concepts.

---

5 Parnas 1972, 1053

6 Ulrich and Eppinger 2012, 185

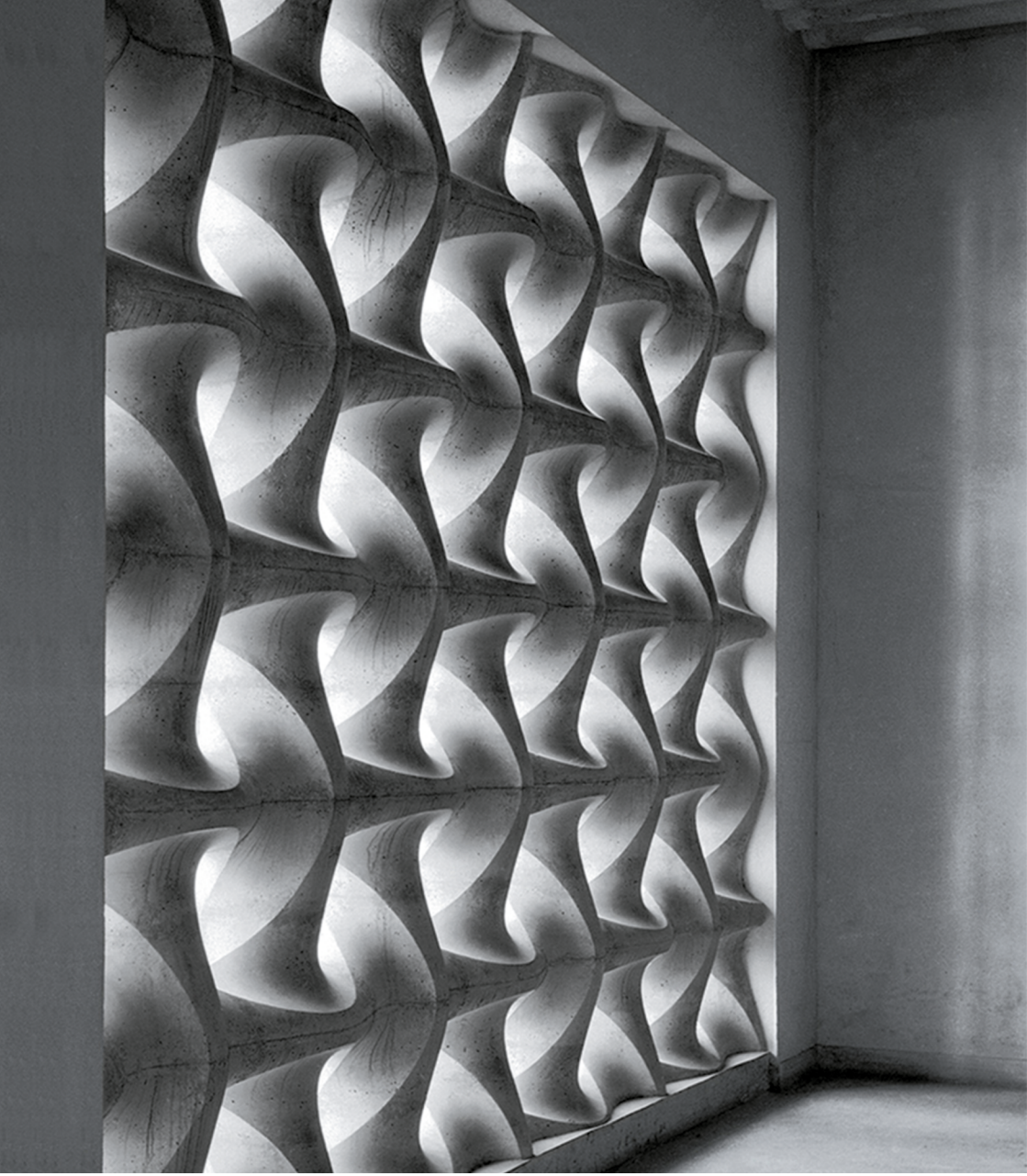


IMAGE 1

The cast-limestone Design 2 (1951) by Erwin Hauer at Vienna's Pfarre Liesing church, 1954.

2.1

Modules, Components, and Systems

Modules are ubiquitous, but the term *module* lacks a standardized definition. The word *module* stems from the Latin word *modulus*, which stands for a small measure. [7–8] The terms *modularity* and *module* are frequently used, for instance, in the product engineering literature. However, despite the popularity of the terms, they are frequently used interchangeably with the terms component and subsystem since they are not standard system engineering terms. [9] There are other fields that address modularity and the definition of modules, such as computer science, design engineering, and architecture. [10]

The definition of the term component is similar to the definition of a module. The Cambridge dictionary defines a component as "a part that combines with other parts to form something bigger." [11] In contrast, the Lexico dictionary defines the term module as "each of a set of standardized parts or independent units that can be used to construct a more complex structure, such as an item of furniture or a building." [12] However, each discipline defines the terms differently based on their contexts. For instance, in software engineering modules can be seen as a set of classes while in the car manufacture industry, a module can be a windshield. Nevertheless, the current literature tends to describe modules as more advanced and more significant than components. [13–14]

---

7 Online Etymology Dictionary n.d.
8 Merriam-Webster n.d.
9 Efatmaneshnik, Shoval and Qiao 2018, 1
10 Gershenson, Prasad and Zhang 2003, 297
11 Cambridge Dictionary n.d.
12 Lexico n.d.
13 Chatras and Giard 2016
14 Ulrich and Eppinger 2012

Despite the challenge of precisely defining modules, different modules share four common characteristics. Firstly, modules are co-operative subsystems that ultimately form a product. Secondly, modules have primary functions within rather than between them. Thirdly, modules have more than one well-defined function that can be individually tested and is created by the components of the module. Fourthly, modules are independent and self-contained units that can be combined with other related units to produce different outcomes.[15]

Finding an unequivocal definition for the term system is challenging because commonly accepted definitions lack precision. For instance, James Miller's definition, "a system is a set of interacting units with relationships among them,"[16] is dubious. Nevertheless, Börje Langefors' definition is vaguer: "a system is a set of entities with relations between them."[17] Alexander Backlund offers a slightly more precise definition. He states that a system must consist of at least two elements and that it is not an aggregate; there must be a connection between the elements.[18] Despite these definitions being similar, there is one clear difference among them: connection types between elements within a system. Miller states that the connection type between elements is interactive, implying that the connection must be functional. In contrast, both Langefors and Backlund do not limit connection types. This is the essential difference since elements within a system can additionally be based on relevance. For instance, one can assume that a bed is a system since it can be formed from different elements by combining a mattress, bedding, pillows, and blankets. In this case, the bed is a system that relies on the relevance between its elements rather than interactions.

15 Russell, Leaney and Botterell 1998, 1
16 Miller 1978, 16
17 Langefors 1995, 55
18 Backlund 2000, 448

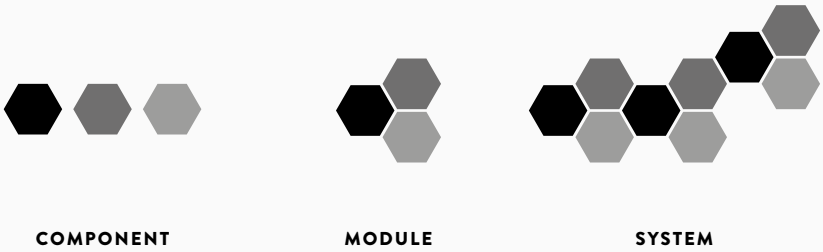


FIGURE 1

A common perception between components, modules, and systems. (Miller 1978; Godbolt 2016)

Any combination with multiple components can be a system; hence, it is safe to say that the definition of a system depends on the context. For instance, a module can be considered as a system since it is a combination of more than one linked component. A system within a system is both a system when inspected closely or a module that is implemented into a larger system when inspected more broadly (see Figure 1).

Individuals do not share the same consensus on how to define, measure, and inspect modularity; thus, comparing two modular systems or products is impractical. It is additionally challenging to prove that one system is more modular than another one. The same applies when inspecting components, modules, and systems—each person has their own approach to measuring and inspecting different systems.[19]

19 Gershenson, Prasad and Zhang 2003, 296

2.2
Modular Product Architecture

Modular product architecture enables significant product development opportunities compared with non-modular architecture. For instance, the modern computer industry is extensively based on modularity. Fast development cycles, affordable consumer prices, and significant innovations are enabled by modular product architecture. However, it is not only the computer industry that has applied the modular design approach in their design stage. A growing number of industries are ready to extend the modular design approach from their production processes to design stages. Modularity is not new in product development. Different industries have applied modularity to the core of their production processes for decades because it has always been easier to manufacture complicated products by dividing the manufacturing process into smaller parts, or modules. Despite this, there are industries that do not exploit modularity in their design stages. [20]

The shift from only using modularity in production processes to additionally covering product design stages is significant. Instead of creating interconnecting physical objects, such as car parts or screws and nuts, creating design rules for a modular system is a challenging task. According to Carliss Baldwin and Kim Clark, a design rule is a privileged parameter that affects other parameters but cannot be changed because if changed, it would cause a costly chain reaction since other parameters would have to adapt to the change. [21] Design rules, similar to architectural decisions, are system-level properties, and thus, cannot be changed easily afterward. [22]

---

20 Baldwin and Clark 1997, 1–5
21 Baldwin and Clark 2000, 68, 75–76
22 Waterman, Noble and Allan 2015, 347

Product architecture is determined by research and development (R&D) teams at the early stage of the product design process. These decisions have a far-reaching impact on both the performance of the company's product and on future product design and development processes. [23] Product architecture defines the core (technical) structure of a product. According to Ron Sanchez, product architecture divides the overall functionalities of a product into specific functional components. Moreover, product architecture defines the interface specifications of how these functional components interact with each other. [24]

The benefits of modular product architecture can be divided into six main categories: product development and design, variance, production, quality, purchasing, and after-sales. In contrast, Kexin highlights three limitations of the modular system. Firstly, designing a modular system compared with a similar interconnected system is a challenging task since the complexity of modular design can be high. Secondly, designers can unintentionally design common units. A common unit module contains an excessive amount of functions and details, causing situations where modules can no longer operate independently. Thirdly, the variants created from the same platform or modules can excessively resemble each other, creating situations where the outcomes are no longer attractive to customers. [25]

Despite modular product architecture having advantages, only a few products are genuinely fully modular. A product that has modular product architecture can be seen as a modular system that has a certain degree of synergistic specificity. Melissa Schilling states that these kinds of systems "might be able to accomplish things that more modular systems cannot," at a cost of recombinability. Thus, synergistic specificity decreases

23 Ulrich 1995, 419
24 Sanchez 2004, 59
25 Kexin 2004, 86–90

the product's modularity. She further forms a general systems theory (see Figure 2) that is built around three main aspects: (1) synergistic specificity, (2) heterogeneity of inputs and demands, and (3) urgency. [26]

The first aspect, synergistic specificity, is the only characteristic that negatively affects the modularity of the product. Despite synergistic specificity allowing greater functionality through the optimization of components, it can establish barriers for others. For instance, it may be challenging to assess or implement components if they are not built with inter-firm modularity in mind. [27]

The second aspect of Schilling's general systems theory, the heterogeneity of inputs and demands, contains the available technological options within a system and the resources and capabilities of the company. The ultimate ambition of a modular system, according to Schilling, is to "enable heterogeneous inputs to be recombined into a variety of heterogeneous configurations." She additionally adds that "the more heterogeneous the inputs are that may be used to compose a system, the more possible configurations there are attainable through the recombinability enabled by modularity." In contrast, the heterogeneity of demands consists of external inputs that impact the modularity of the product. Heterogenous demands determine what kind of elements the modular system should provide in order to answer the established needs. The higher the diversity of (technological) options available, the more attractive the system both for customers and producers. [28]

The third aspect, urgency, is created by the speed of technological changes and competitive intensity. Schilling states that modular solutions are highly attractive options since they can rapidly adopt the latest changes and trends and therefore respond to heterogeneous demands

26 Schilling 2000
27 Schilling 2000, 316–21
28 Schilling 2000, 217–23

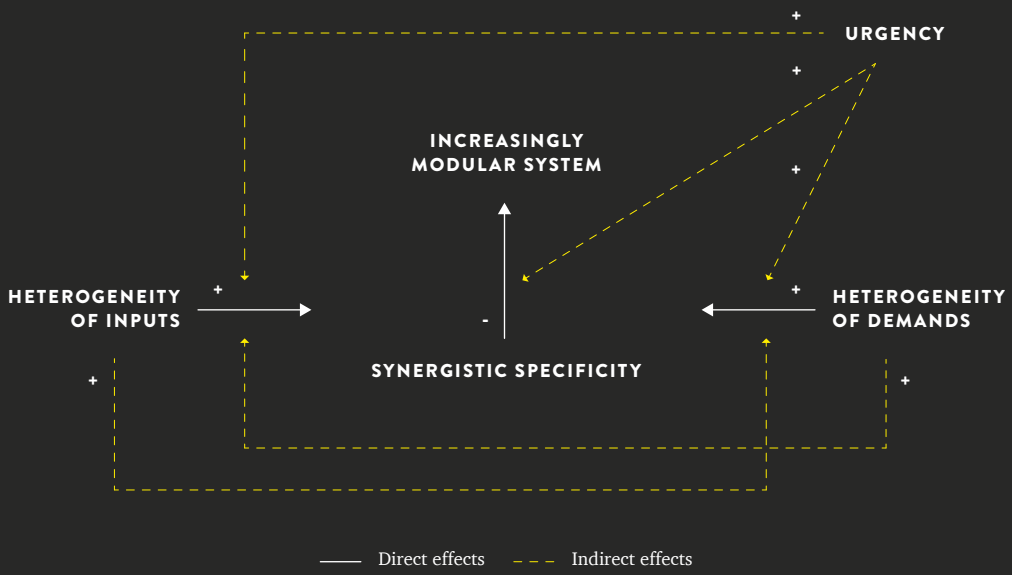


FIGURE 2

A general theory of modular systems adapted from Melissa Schilling's work (2000).

more swiftly than a non-modular product. Lastly, she notes that from the customer's point of view, modularity reduces costs and enables customers to upgrade particular components as new options become available. [29]

As previously discussed, modular product architecture both enables rapid product development and reduces production costs. Therefore, defining product architecture is a crucial part of industrial product development activities. Ron Sanchez and Robert Collins point out that companies may pursue either closed-system or open-system strategies when creating modular architecture. In the closed-system strategy, the product is built from components supplied in-house. In contrast, a product based on the open-system strategy offers interface specifications to other companies, allowing outside companies to develop components for the product. In addition to these two strategies, a company can establish new industry standards in collaboration with other companies. [30] However, a product can additionally be somewhere between the open-system and closed-system strategies.

From a business perspective, modular product architecture increases opportunities to test different options through product variations, and hence, it increases company competitiveness. [31] Industries have become more competitive when product life cycles have shortened, creating a more demanding environment. Through strategic planning, a company can utilize the same components across projects. One valuable benefit of modular product architecture is the ability to rapidly configure new product variations at low cost by mixing and matching components. [32] Without the benefits of modular product architecture, companies can face difficulties when they cannot react swiftly and efficiently to the growing needs of customers.

---

29 Schilling 2000, 326–28
30 Sanchez and Collins 2001, 648
31 Nobeoka and Cusumano 1997, 169–70
32 Sanchez and Collins 2001, 646

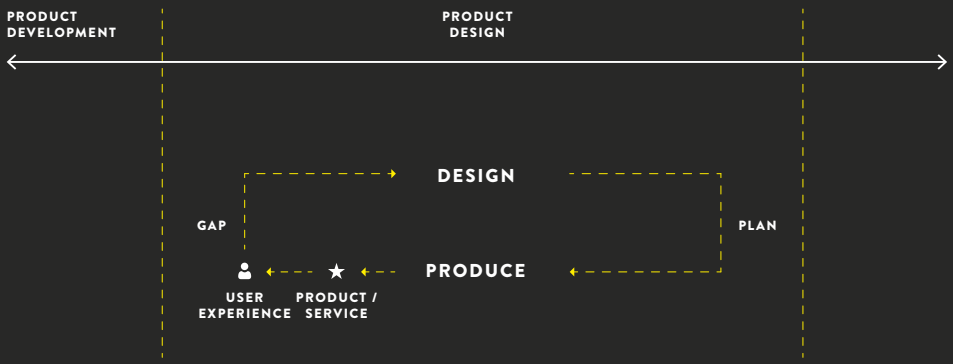


FIGURE 3

"Design and production are the two activities that deliver artifacts to address gaps in the user experience" — Karl Ulrich. Figure adapted from Ulrich's work (2011).

2.3

Elements of Product Design

Product design is a process by which a product is conceptualized, designed, and created (see Figure 3). Product design is a part of product development, which is a broad concept that contains a complete product development cycle from market analysis to logistics.

Product design, as well as product designers, blurs the boundaries among disciplines. Briefly stated, product design concerns the enrichment of quality of life, but it additionally has a commercial perspective; in order to attract new customers, the product must be appealing and stand out from competitors' products.[33]

33 Rodgers and Milton 2011, 7

The definition of the term product design has its challenges since the term is used in numerous ways. According to Karl Ulrich, the term has regional varieties. Product design on the West Coast of the United States—more precisely, in Silicon Valley—frequently refers to the activity of making forms created by industrial designers into production-ready plans. In contrast, on the East Coast, the term is more synonymously used with industrial design. He additionally defines the term as "...conceiving and giving form to goods and services that address needs." [34]

Despite the elusiveness of the term's definition, one commonly repeated argument appears in the academic literature: product design is a process during the life cycle of the product where the product is defined and produced to the masses. This is the common thread in the current academic literature related to product design. Both modularity and product design are common themes in engineering, economics, and business but conspicuously absent in the (visual) arts. For instance, based on the available literature, all artistic productions, such as paintings and illustrations, are not considered as products since the artistic pieces are generally neither mass-produced nor follow the universal principles of ideal design.

The literature offers numerous process models and frameworks to address product design. For instance, the BAH model created by and named after Edwin Booz, James Allen, and Carl Hamilton, [35] the lean startup approach introduced by Eric Ries, [36] and the stage-gate model (or phase-gate model) by Robert Cooper [37] are well-used frameworks for product design process management. Product design methodologies generally have similar elements and nearly identical structures. These

34 Ulrich 2011, 394
35 Booz, Allen and Hamilton 1982
36 Ries 2011
37 Cooper 1986

include an initial research and strategy phase, then a design and development phase, then a product launch phase followed by potential post-launch activities. Nonetheless, Ries' lean startup approach offers a suitable framework for small companies since its focus relies on product development and rapid learning through constant measuring instead of time-consuming research and analysis.

CHAPTER III

Designing Modular Products

This chapter inspects modularity in artistic and digital product design processes. The literature used throughout this chapter is based on expert insights and ideas, design handbooks, and curated internet sources.

A design system is a digital, modular, and scalable system that has been built to control the visual consistency of a product in a rapidly shifting environment. Production teams behind digital products are continually growing and reforming; new team members join, and others may leave. Freelancers and external parties vary throughout the production process, and the product itself may evolve in a direction that requires new hires. A design system offers a mutual language among team members, both within a company and among external stakeholders. In a digital and web-based world, a design system can cover the entire product management, development, and design phases within a company (see Figure 4). In other words, design systems are solutions to establish a shared understanding between team members when designing and building sophisticated web-based products.

Despite the term design system having been a buzzword of the late 2010s,[38–39] its definition remains vague. For example, Rune Madsen describes the phenomenon as "a philosophy that encourages designers to define the rules of their designs as a system of instructions that can be used on more than a single product."[40] In contrast, Alla Kholmatova offers a practical definition of the term: "a design system is a set of interconnected patterns and shared practices coherently organized to serve the purpose of a digital product."[41] Marco Suarez's point of view is similar: "[a design system] reduces design debt, accelerates the design process, and builds bridges between teams working in concert to bring products to life."[42] These varying viewpoints indicate that it is difficult to precisely define the term since each discipline perceives differently. For

38 Awwwards 2019
39 Cao and Cousins 2018
40 Madsen 2018
41 Kholmatova 2017, 22
42 Suarez et al. 2017, 2

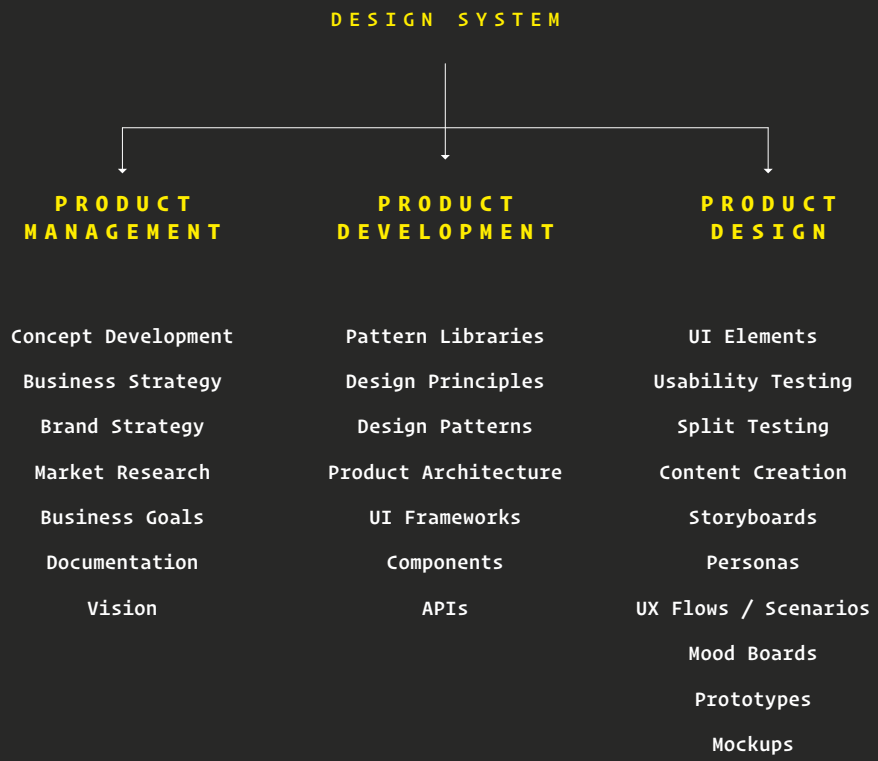


FIGURE 4

Each design system is a unique entity, and hence, their structures vary between companies. A design system is ultimately a living, evolving system that can cover any topic and theme, from product management and development to design.

instance, software developers may see design systems as snippets and component libraries whereas UI designers can perceive design systems as visual guidelines. Moreover, the term can be incorrectly interchanged with the terms *pattern library*, *modular design framework*, *component design*, *design language*, and *atomic design*. [43]

3.1

A Brief History of Design Systems

The foundation for design systems lies in a modular software architecture that has been around since the 1960s. Design systems are an answer to the software crisis (additionally known as the software gap), a term that was introduced at the first NATO Software Engineering Conference in 1968. [44] A few years later, Edsger Dijkstra investigated the problem in his article "The Humble Programmer." He explains how sophisticated systems have become more complex to produce since the rapid development of computers: "as long as there were no machines, programming became a mild problem; when we had a few weak computers, programming has become an equally gigantic problem." [45]

Presently, the web industry faces similar problems to those of the software industry in the 1960s. The wide variety of web technologies, design standards, operating systems, and smart devices—such as smartphones and tablets—set requirements for demanding and comprehensive solutions. At the same NATO conference in 1968, Douglas McIlroy introduced one solution to tackle this problem in his work "Mass Produced Software Components: the Component-Based Approach to Software Development." He argued that the foundation of the software

43 Rutherford 2018
44 McClure 2001, 70–72
45 Dijkstra 1972, 860–61

industry was not well-founded since it was not industrialized and standardized. He additionally noted that it did not make sense to reinvent and repetitively rewrite the same basic software components and that a selection of mass-produced components should be available to use in different applications. [46]

Over five decades later, sophisticated systems facilitate the same component-based approach that was introduced in the late 1960s. Currently, however, the design industry faces similar problems; digital and online-based services have evolved alongside web technologies. Darcy DiNucci branded this transformation as *Web 2.0* in her article "*Fragmented Future*" in 1999. She addressed how technical variances, such as different screen sizes, input methods, and connection speeds, in addition to the lack of standardization, would divide the web publishing industry into smaller fragments in order to tackle the future design and software development challenges. [47]

Building a design system or defining modular design architecture cannot be possible without standards and shared best practices in web development. Today, both designers and developers have a sophisticated ecosystem that supports agile web development and sets standards and best practices for design. Building a design system without using open source web standards or frameworks would be difficult since it would require increased customization work, design and code-wise.

As stated in Section 2.2, building sophisticated products is manageable and feasible when the product is divisible into smaller parts, that is, into modules and components. In other words, a design system and modular design architecture are comparable with old-fashioned product development processes for manufacturing modular goods—but in this case, the processes are meant for building products for digital platforms.

46 McClure 2001, 79–85
47 DiNucci 1999, 32

3.2

Standardizing Web Development

Design systems have been established for web-based products, and thus, they follow the latest web standards set by the web industry. Therefore, the same design system is not suitable to be used both in Web and native app development. As the previous section concluded, in order to build modular products by mixing and matching components, there must be industry-wide standards.

As discussed in Section 2.2, an open-system product architecture strategy does not work well if the used components are not standardized. Since design systems are highly context-based and company-specific, they generally follow the closed-system strategy. However, design systems are closed entities, implying that they are not built with external modularity in mind. Thus, they do not generally support mixing and matching components among different design systems.

The online industry has frequently sought to set and define common ground and standards for web development. Defining and setting web standards has been an ever-changing process since 1989 when Tim Berners-Lee invented the World Wide Web. In 1994, five years after the birth of the Web, Berners-Lee founded the World Wide Web Consortium (W3C) at the Massachusetts Institute of Technology (MIT) in collaboration with the European Organization for Nuclear Research (CERN) to address these issues. [48]

The web industry has changed and evolved over the past three decades; web technologies and best practices iterate when new and more sophisticated solutions arise. For instance, Adobe's Flash (ActionScript) was widely used to create interactive content for websites. Today, this

48 World Wide Web Consortium (W3C) n.d.

web technology has been surpassed by the development of newer and completely open web standards (OWS), such as HTML5 and WebGL. [49] However, several web technologies became virtually nonexistent until a sudden rise in popularity. For example, one of these technologies is the Scalable Vector Graphics format (SVG) that was developed by W3C and published in 1999. Doug Schepers pointed out three main reasons why it took approximately a decade before SVG became a standard in web development during his interview "SVG with Doug Schepers" by Jen Simmons. [50]

Firstly, the SVG format was not supported by Microsoft's Internet Explorer (IE) 7 and IE8 browsers when the browsers dominated the browser markets. Secondly, internet browsers relied on third-party plugins for rendering both Flash and SVG formats in the 2000s. SVG used the SVG viewer plugin that was developed and distributed by Adobe. It was unsurprising when Adobe stopped developing the plugin after the company acquired Macromedia (the original owner and developer of Flash) in order to promote Flash. Thirdly, the rise of the smartphone era set new requirements for web standards when the internet became accessible through modern smartphones. Simultaneously, plugin web technologies became obsolete since mobile browsers no longer supported them. [51] In fact, Flash was not the only technology to face this destiny; Microsoft attempted to push its Silverlight web technology to rival Adobe's Flash with the same result. [52]

The latest changes in the web industry can be traced to a single moment that permanently changed the (mobile) web development and entire web industry: the rise of the Apple era. It was clear that plugin-

49 Adobe Corporate Communications 2017
50 Siegler 2010
51 Schepers 2014
52 Siegler 2010

based web technologies, such as Flash and Silverlight, would become obsolete after Apple's former CEO and co-founder Steve Jobs published his letter "Thoughts on Flash" in 2010. He stated that Apple would only support open web technologies in the future and highlighted how web technologies that required development tools owned and supplied by third parties were precarious for the development community. [53] It was therefore foreseen that modern web development would be built around web technologies defined by standards organizations rather than companies. In other words, this is one of the reasons why modern websites and design systems use open source technologies such as HTML, CSS, JavaScript (JS), Hypertext Preprocessor (PHP) and Structured Query Language (SQL), and more.

Image-wise, images used on the web are either raster (lossy) or vector (lossless) images. Raster-based image formats, such as Portable Network Graphics (PNG) and Joint Photographic Experts Group (JPEG) are based on pixels. In contrast, vector graphics are mathematical formulas consisting of numerous graphical primitives and attributes on a grid. Such primitives are points, lines, curves, and polygons that are enhanced with different attributes, such as fill and stroke. [54] The International Standards Organization (ISO) and Telephone Consultative Committee (CCITT) established the JPEG format in 1986 for videotext services. Videotext services were global and text-based information services delivered over analog telephone lines by the leading telecommunication service providers in the early 1980s. The JPEG format (more precisely, a compression technique) was developed so the original images could be compressed, and thus, delivered more rapidly via telephone lines. [55]

53 Jobs 2010
54 Rosenbaum and Tominski 2003, 3–4
55 Hudson, et al. 2017, 96–98

3.3

Creating Modular Design Architecture

The opportunity to create modular design architecture for digital platforms became possible as a result of the standardization of software during the 1990s. Nevertheless, as already discussed in Section 2.1, one module is not sufficient to build a system; a system is a modular entity that consists of interconnected modules. This section focuses on creating modular design architecture for digital products. In this context, the term platform is more suitable since web-based products are generally considered to be platforms.

The majority of platforms are structurally nested systems; smaller systems of components form more complex systems of components. Hierarchically, the lowest system of components within a more extensive system is called an elementary subsystem, where the smallest unit is called an elementary particle. There are no rules on how to determine what subsystem is elementary since each discipline and project has its own concept of what the elementary particle is.[56] In the context of design systems, the elementary subsystem can be a set of color values, where one color value is an elementary particle. It can additionally be something abstract such as a design rule where the color space is monochromatic and based on a specific shade of blue.

The interconnected subsystems that form the platform are known as modules. Modules can be designed independently, but they collectively work together. Different industries have applied a modular approach to their product design processes, and the current trend is catching up with the design industry.[57] As previously discussed in Section 2.1, modularity, in essence, indicates that one module from a system can be separated

56 Simon 1962, 478
57 Baldwin and Clark 1997

from its original context, recombined and reused elsewhere if the design pattern (see Section 3.4) is the same.

In order to design modular, component-based products, designers need to have comprehensive knowledge of different design applications and tools that support modular design approaches. [58] Today, there are several design applications available that support modular design architecture to some degree, such as Sketch by Bohemian Coding[i], Adobe's Adobe XD[ii], InVision Studio by InVisionApp Inc.[iii], and Figma by Figma, Inc.[iv], to name a few.

Modularity is the foundation of a design system since a design system is a consistent system that covers themes and topics from product management, development, and design. In contrast to entirely modular systems, a design system is bound to its original context. For instance, one could build a speaker system by combining different types of speakers manufactured by different companies since all speakers share the same standardized input and output interfaces. However, in the context of a design system, one generally cannot take one customized module from a random design system and immediately implement it without further customization. Therefore, design systems generally follow a closed-system strategy and have a high level of synergistic specificity (see Section 2.2).

i www.sketch.com
ii www.adobe.com/products/xd.html
iii www.invisionapp.com/studio
iv www.figma.com

58 Curtis 2010, 79

Design systems are an ideal example of a modular system with a high degree of synergistic specificity since they are typically only designed and produced for internal use. All modules and components within a design system are based on product-specific visual guidelines and are optimized to work with each other using the same unit system and grids. The modules and components within a design system are additionally highly polished—both technically and design-wise.

This thesis mainly focuses on the product design phase since it is bound to the product's modular design architecture. As previously discussed at the beginning of this chapter, a design system is an entity that can consist of any phases and assets related to product management, development, and design. Hence, the thesis additionally briefly covers tasks and phases related to product management and development.

3.4
Foundation of a Design System

Despite each design system being a unique entity, all design systems share recognizable characteristics. When we look at a few well-known web-based design systems, such as Google's Material Design,[59] IBM's Carbon Design System,[60] and GitHub's Primer,[61] it is evident that they are formations of various smaller systems. As previously stated in Section 2.2, similar to any modular system, design systems can be partitioned into smaller parts that can be individually maintained. This thesis presents four topics that are frequently covered in the current literature on design systems; (1) visual language, (2) pattern and component libraries, (3) source code, and (4) documentation.

59 Google LLC n.d.
60 IBM n.d.
61 GitHub n.d.

Visual Language

A visual language is the visual appearance of a product that is generally strictly based on visual guidelines set by the product's brand identity. There are numerous definitions of visual language. For instance, Robert Horn defines the concept as a communication tool that is an integration of words and visual elements distinguished from natural languages by its characteristics.[62] According to Horn, the visual language is based on the tight integration of words and visual elements; it is a platform for both designers and developers through which they deliver content and messages for end-users. However, this vision is ambiguous since, for instance, sign languages are based on visual cues rather than on spoken words.

Users see and interact with different visual languages when they look at the layouts or views of the product; they consciously and unconsciously recognize specific patterns, groups, and principles. A layout—or other graphical element—is a message to be understood and interacted with by its viewer. Different attributes affect how people perceive and decrypt that message. For instance, age, educational background, language, and culture are factors that can create a bias towards the original intention of the designer.[63]

Visual languages generally follow commonly used user interface elements and trends. Since it is difficult to know how end-users will understand and perceive different visual stimuli, it is beneficial to apply design patterns and UI elements that are commonly known among end-users. These commonly accepted UI patterns are additionally addressed as user experience (UX) principles. Presently, there are over one hundred

62 Horn 2001, 1–3
63 Malamed 2009, 20–22

UX principles that enhance usability.[64] For instance, red generally implies errors while green implies successful actions, the size difference of the text between titles and paragraphs displays text hierarchy, and the contrast between background color and text improves readability.

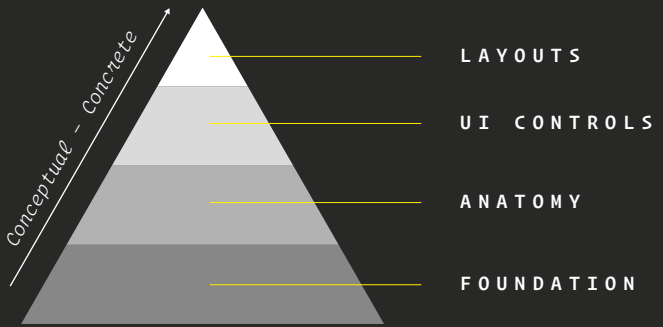
A visual language of a design system can be inspected using three overlapping layers; foundation, anatomy, and UI controls (see Figure 5).[65] As with any architectural system, the foundation of a visual language should be established before defining the visual language in detail, as additions are built on top of this foundation.

The foundation of a visual language is a set of internal rules and goals that frames an ideal self-portrait of the company and its products, as illustrated in Figure 5. Despite the fact that the foundation of a visual language is hidden from end-users, what is built on top of it is visible. For example, the anatomy of a visual language holds essential settings and rules that define the UI elements of the product (UI controls), such as buttons, forms, tabs, and headers. Therefore, both designers and developers must follow the rules set in the foundation of a visual language to achieve consistency throughout the product. A consistent visual language maintains effective and intuitive communication between the product and its end-users.[66]

Component and Pattern Libraries

A component library is a collection of user interface elements that share the same consistent visual language and can be used multiple times throughout the product.[67] The main difference between a component library and UI controls (as presented in Figure 5) is the degree of abstractness. The

64 Lidwell, Holden and Butler 2010
65 Gonzalez 2017
66 Kuznetsov 2020
67 Suarez et al. 2017, 67–69



LAYOUTS

Templates Elements Sections
Menus Widgets Forms CTA

UI CONTROLS

Buttons Headers Breadcrumbs
Tabs Toggles Date Picker
Progress Bar Input Fields

ANATOMY

Color Space Typography
Shadows Gradients Depth
Depth Motion Alignments

FOUNDATION

Brand Identity Brand Strategy
Design Principles UX Strategy
Business Goals Vision

FIGURE 5

A typical structure and content of a visual language based on Gonzalez's post (2017).

figure presents theoretical UI elements that guide developers in generating functional UI assets. In contrast, a component library consists of ready-to-use assets that can be directly implemented into a product. However, the term is occasionally confounded with the term *pattern libraries*.

Component and pattern libraries share similar characteristics: they are libraries and their core ideals are the enhancement of reusability, consistency, and productivity. Although component and pattern libraries are highly related and can be used interchangeably in specific contexts, a pattern is an abstract and universal solution to a global design problem. Christopher Alexander highlights that "each pattern describes a problem which occurs over and over again in our environment, and then describes the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice." [68] In contrast, a component within a product-specific component library is a reusable asset that is a precise solution to a specific design problem. [69]

To summarize, a component from a component library can define, for example, how a login button looks and behaves. In contrast, a pattern from a pattern library can define the entire user flow for the login process. Therefore, components are highly polished and defined assets for particular context and usage, whereas patterns are not used in this manner. [70]

Source Code

The file architecture of a design system is generally a codebase that consists of different files and folders that follow a well-designed structure. Despite the existence of numerous correct ways to build design systems, their management and development are typically made through git repositories. [71]

---

68 Alexander, Ishikawa and Silverstein 1977, X
69 Curtis 2010, 174
70 Curtis 2017
71 Pate 2017

The Git—as we know it today—has been widely used since it was first created by Linus Torvalds and published in 2005. Managing a design system through a git repository (repo) offers rapid product development speed and data integrity. [72] It allows non-linear workflows, which are mandatory when it comes to building modular products and design architectures. However, Git was not the first of its kind. Before Git, there were, for example, Concurrent Version System (CVS) originally coined by Dick Grune, BitKeeper, and Subversion, along with others. [73] These systems share similar characteristics with Torvalds' git system. Git is a distributed version control system (DVCS) for committing, commenting, and tracking changes made in source code, and it is designed for organizing software development work through branching. [74] However, Git can track changes in any set of files.

Each module, component, and element within a design system has a unique code snippet that is reusable throughout the product. For instance, a typical HTML or CSS code snippet has two parts (see Figure 6). The first part, a selector, selects the HTML element(s) on which a set of CSS rules are to be applied. The second part, declarations, consists of properties and values. [75] For instance, in web development, an HTML file contains the page structure where each element has an HTML tag that is usually enhanced with a CSS class. If the same selector is defined more than once in source code, the outcome is a combination of different declarations that can cause the unpredictable behavior of an HTML element.

The way a UI component appears and behaves originates from the product's source code. The more the product utilizes the same CSS classes, the more consistent and more straightforward it is to maintain. In order to

72 Chacon and Straub 2014, 13
73 Fuller 2008, 64–65
74 Chacon and Straub 2014, 11–12, 62
75 W3Schools n.d.

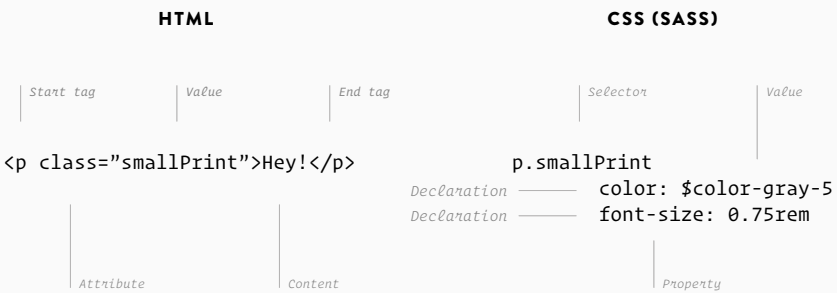


FIGURE 6

A simple code structure has two parts.
Generally, code snippets are formed using HTML and CSS.

avoid rewriting code, it is essential to build a clear and consistent structure and naming convention for the source code. Micah Godbolt summarizes this as follows: "Start off with bad markup, and you'll be writing bad CSS and bad JavaScript to make up for it. Start with great markup, and you'll be able to write more scalable and maintainable CSS and JavaScript."[76]

Documentation and Naming Convention

Consistent and self-explanatory naming conventions play a central role when building robust design systems. However, there are as many different ways to manage naming conventions as there are products or systems within a product. Therefore, the method of naming assets and snippets—be that based on underscores or camelCase—is not that important if stakeholders understand them.[77]

Technically, a naming convention consists of a set of rules for choosing the character sequence to be applied for class identifiers,

76 Godbolt 2016, 23
77 Baldwin 2020

which indicate variables, functions, and other entities in source code and documentation. A consistent and justified way of naming classes and variables throughout the source code and documentation of a design system enhances the usability of the code and documentation and makes the code understandable for all stakeholders. [78] For instance, a class name "button-primary" implies that the element is a link or button and that it is, in fact, a primary button. In contrast, a class name "1-btn" does not explain its use case as unequivocally. The importance of the naming convention is particularly underlined when there are dozens of buttons within a design system.

A consistent and description-based naming convention supports other best practices found in software development, such as the single source of truth (SSoT) and the don't repeat yourself (DRY) concepts. The concept of a design system is similar to the SSoT concept; both aim to have one unified and consistent source of information that is up-to-date, robust, and flexible for future development. Design systems are similar to the degree that a design system can be described as a "single source of truth" in the professional literature around design systems. [79]

Another best practice found in software development that can be applied to design systems is the avoidance of information repetition and code duplication. DRY is a principle that aims to eliminate the need for repeatedly rewriting the same code—hence the name of the concept. [80] Hunt and Thomas summarize its core message as follows; "Every piece of knowledge must have a single, unambiguous, authoritative representation within a system." [81] Both concepts—SSoT and DRY—are achievable through clear naming conventions, both in the product's source code and in its documentation.

---

78 Greenberg 2019
79 Hacq 2018
80 Foote 2014, 165
81 Hunt and Thomas 1999, 27

CHAPTER IV

Research Methods and Approach

This chapter presents the research methodology of this thesis and its data collection and analysis methods.

The first research question of this thesis is as follows: "How can modular design architecture accelerate product design processes when creating sophisticated web-based products?" In order to find this answer, this thesis applied the current academic and professional literature to a study that scrutinized three web-based streaming platforms: Deezer, Tidal, and SoundCloud.

Web-based streaming services were selected for two reasons. Firstly, they share comparable design patterns, user flows, and functions. Therefore, finding common denominators among highly comparable services is more feasible than among entirely different web-based products. Secondly, design systems are solely built for sophisticated web-based products. Fortunately, web-based products are convenient to study since their design and product architectures can be extensively inspected in their source codes. For example, naming conventions, UI frameworks, JavaScript libraries, and ready-to-use design systems—such as Mineral UI and Primer—can be scrutinized by skimming through source codes. Topics and functions related to backend techniques and solutions are not as visible, but on the other hand, design systems are not commonly perceived to cover topics and solutions related to backend development.

This thesis conducted a study that had two primary research phases. The first research phase was a benchmarking process that scrutinized the selected streaming services. In contrast, the second research phase was a data coding and analysis process based on the data acquired from the first research phase. The second research phase was conducted by applying the principles and frameworks found in the grounded theory research methodology.

The grounded theory methodology encourages researchers to seek and form new theories based on research data instead of conducting studies according to another researcher's theory, framework, or research process. Therefore, grounded theory fits well in the context of this

thesis, since the phenomenon behind this thesis is modern, and there has been limited academic research work done on the topic. The outcome of this study might have been predictable, bringing limited or no new information, if a prevailing theory related to design systems, such as the atomic design approach, had been applied to the research processes of this thesis.

The outcome of the first research question provided the assets and insights required to address the second research question: "How can modular design architecture be defined when designing new web-based products?" This thesis put the established framework—the framework of architectural dependencies within a modular design architecture—into practice to find an answer to the second research question.

This chapter starts by introducing the grounded theory methodology. The methodology is introduced from the perspective of this thesis; it, hence, excludes methods and frameworks that are irrelevant. For example, qualitative interviewing was excluded from this thesis since no interviews were conducted during the study. The chapter's second section presents an introduction to benchmarking, providing information on how to identify essential properties and characteristics of what to benchmark when creating robust web-based products from the ground up.

4.1
Introducing Grounded Theory

Barney Glaser and Anselm Strauss pioneered the grounded theory methodology in the 1960s during their studies of terminally ill patients in American hospitals. Their studies formed systematic methodological strategies that could be adapted by other scientists for studying other topics as they worked on their analyses.[82] They presented grounded theory strategies and methods, such as theoretical sampling[83] and the constant comparative method,[84] in more detail in their book *The Discovery of Grounded Theory* in 1967.

Grounded theory has a few core elements that distinguish it. For instance, data collection and its analysis phases happen concurrently throughout the project (the constant comparative method) and analytic processes advocate to discover and develop new theories based on the data rather than verify and adapt to existing models. Grounded theory is an inductive research approach: its purpose is to form new ideas during the research process (observation—theory) rather than test hypotheses (theory—confirmation).[85] The discovery of the final theory from acquired data is achieved by systematically identifying, developing, and temporarily testing different varieties of the same theory throughout the continuous process of data collection and analysis.

Kathy Charmaz coined an overall process model for a grounded theory project in her book *Constructing Grounded Theory: A Practical Guide through Qualitative Analysis* (see Figure 7). The process figure illustrates the process in linear form, but Charmaz underlines that in practice, the

82 Charmaz 2006, 4
83 Glaser and Strauss 1967, 45
84 Glaser and Strauss 1967, 102
85 Gibbs 2010

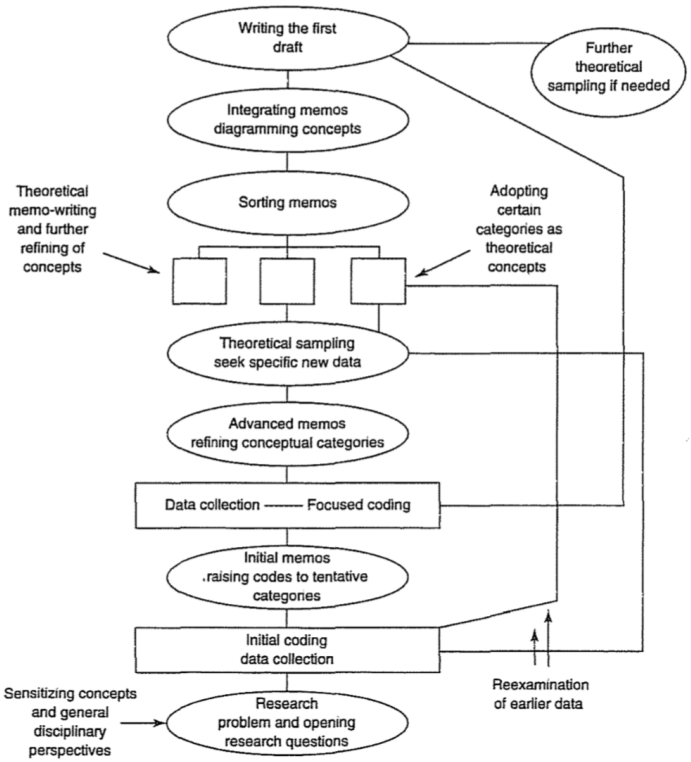


FIGURE 7

A grounded theory process model by Kathy Charmaz's figure (2006).

project is not a linear process. Occasionally—and particularly when a celebrated idea appears—researchers may return to the field to acquire in-depth knowledge, despite a project being near completion.[86] A typical grounded theory study has, at a minimum: (1) data collection, (2) memo-writing, (3) data coding and analysis, and (4) iteration between the first three phases.

86 Charmaz 2006, 10–11

Data collection is the foundation of every research project. Data collection methods vary depending on the type of research—qualitative research applies different methods than quantitative research, and there are numerous methods for the acquisition of data. Some methods can be both qualitative and quantitative, such as benchmarking. Particularly in a grounded theory study, the acquired data are crucial to the research since the data form a foundation for the entire research process, guiding it from the beginning. Hence, the credibility of research is based on the collected data. [87]

Memo-writing is an essential part of a grounded theory project, and it should start from the beginning of the research and continue to the end since its quality grows together with the research itself. [88] Memo-writing is a process whereby researchers write personal and informal analytical memos throughout the project. Researchers compare memos from various research stages to gain insights and fresh ideas on the research topic. [89] Written memos can consist of highly abstract sketches accompanied by a few words that are not self-describing to the readers.

Data coding and analysis are fundamental processes in grounded theory research. There are three basic types of data coding found in grounded theory: open, axial, and selective. [90] John Creswell and Cheryl Poth summarize these three basic data coding approaches: "Grounded theory provides a procedure for developing categories of information (open coding), interconnecting the categories (axial coding), building a 'story' that connect the categories (selective coding) and ending with a discursive set of theoretical propositions." [91] Open coding can be initial or focused. Initial

87 Charmaz 2006, 18
88 Corbin and Strauss 1990, 10
89 Charmaz 2006, 80–82
90 Corbin and Strauss 1990, 12
91 Creswell and Poth 2018, 203

coding helps to closely analyze the collected data while focused coding allows researchers to separate, sort, and synthesize a vast amount of data. [92] However, these two coding methods are not mutually exclusive. Initial coding is the first major phase in a coding process, and it is not as selective and conceptual as the second phase in a coding process (focused coding).

Kathy Charmaz presents three approaches to conduct an open coding process. The first approach is word-by-word coding, where each individual word—or in the context of this thesis, any similar simple data—is analyzed. The second approach, line-by-line coding, provides more distance to the code than the first approach. The third approach, incident-by-incident coding, focuses on comparing incidents with each other rather than meticulously inspecting data. Despite moving from initial coding to focused coding, the process is entirely linear. Focused coding analyzes the outcomes of initial coding. Selecting the coding approach depends on the type of data collected, its level of abstraction, the stage of the research process, and the purpose of collecting the data. [93]

The second basic coding type, axial coding, investigates relations between categories and their subcategories, and these discovered relations are then examined against the data. [94] According to Charmaz, "[axial coding] specifies the properties and dimensions of a category, and reassembles the data you have fractured during initial coding to give coherence to the emerging analysis." [95]

The final basic coding type, selective coding, frequently occurs after the first two coding phases, during the later phases of a grounded theory study (see Figures 7 and 8). The goal of selective coding is to form a core category that covers the other categories. Hence, there are

92 Charmaz 2006, 11
93 Charmaz 2006, 53–58
94 Corbin and Strauss 1990, 13
95 Charmaz 2006, 60

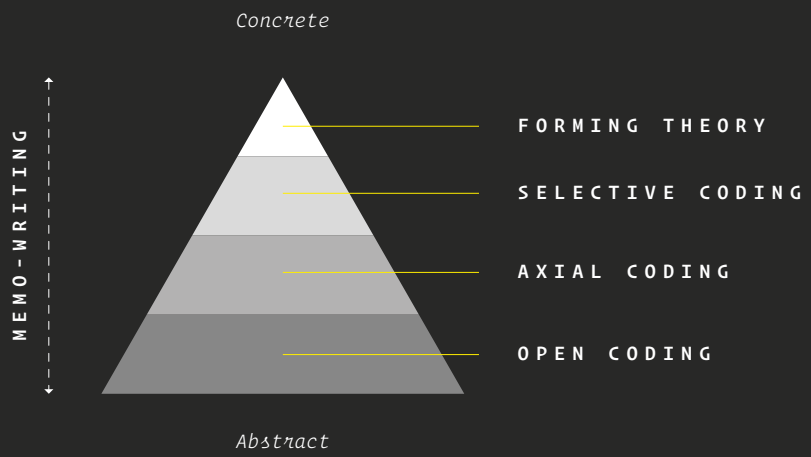


FIGURE 8

Different coding phases (abstract to concrete) and methods applied in grounded theory.

frequently connections between the core category and other categories around it since the core category outlines the central phenomenon of the study.[96] In other words, selective coding creates a single storyline for the study by establishing a common thread between the categories developed over the open coding phase.

Despite grounded theory having been first introduced in social sciences, it has been widely applied to other disciplines, such as in business and software development.[97–98] This thesis applied data coding methods found in grounded theory to examine modularity and architectural

96 Corbin and Strauss 1990, 14
97 O'Reilly, Paper and Marx 2012
98 Waterman, Noble and Allan 2015

elements within web-based products and to coin a new framework based on the acquired data.

It is essential to underline that although the coding methods found in grounded theory offer suitable tools to analyze and categorize computer languages and acquired structured data, the coding methods were originally meant for coding and analyzing natural languages and unstructured text. Studies, however, demonstrate that grounded theory is an increasingly popular research method in software engineering, despite its known challenges, such as managing large amounts of heterogeneous data and coding unconventional texts. [99] Moreover, this thesis applied open coding methods to categorize several computer languages, such as HTML, CSS, and JavaScript.

Grounded theory, similar to any other research methodology, has advantages and disadvantages. Grounded theory can be criticized for its high learning curve as open coding is a time-consuming process. [100] It is additionally criticized because researchers frequently write a literature review after the research is completed. [101] Writing a literature review after completing the analysis process to avoid bias in the findings can be difficult if the prior knowledge around the research topic is limited or the structure of the study is unclear. Therefore, a literature review can be seen as an opportunity to "set the stage for what you do in subsequent sections or chapters," as Charmaz describes. [102] Grounded theory studies additionally tend to have limited generalizability because research findings, new ideas, and established theories are bound to their original data and context. Moreover, research findings of grounded theory research are frequently heavily influenced by the researcher(s).

---

99 Stol, Ralph and Fitzgerald 2016
100 Myers 2020, 134
101 Charmaz 1990, 1163
102 Charmaz 2006, 166

4.2
The Power of Benchmarking

Benchmarking is the process of identifying, understanding, and adapting best practices from other organizations to acquire insights on improving the performance of a product or service.[103] The term lacks a precise definition but generally covers topics related to measurement, discussion, comparison, identification of best practices, implementation, and improvement of a product or service.[104] Robert Camp's definition is frequently cited in the academic literature: "Benchmarking is the search for industry best practices that lead to superior performance." He states that benchmarking should be a discovery process and learning experience rather than a tool for discovering resource reductions.[105] Moreover, there are two fundamental types of benchmarking: internal and external benchmarking. Internal benchmarking refers to the comparison of the organizational performance internally, whereas external benchmarking refers to the comparison of global performance against competitors within the same industry.[106]

Each discipline has its own variation of benchmarking processes and their goals. For instance, in the business world, there are three types of benchmarking: strategic, performance, and process benchmarking.[107] However, these three categories are umbrella terms. They can be broken down into specific approaches and analysis methods, such as financial, performance, metrics, strategic, and personnel benchmarking, to name a few.[108]

---

103 Kumar, Antony and Dhakar 2006, 294
104 Anand and Kodali 2008, 258
105 Camp 1989, 12–14
106 Anand and Kodali 2008, 266
107 Bogan and English 1994, 7–9
108 Patel and Kleiner 2017, 28–29

Strategic benchmarking is applied when the goal is to identify the ideal way to compete in the market. During the process, the company identifies the winning strategies—within or outside of its industry—that successful companies use and apply them to their own strategic process.

Performance benchmarking is applied when comparing a product or service against a similar product or service of a competitor. However, performance benchmarking can additionally be applied to any measurable metric. The goal of a performance benchmarking process is to discover how robust a service or product is when compared with competitors.

Process benchmarking is applied when the goal is to refine and improve internal processes within a company by identifying and applying best practices that competitors use. Process benchmarking is generally a continuum for performance benchmarking since a performance benchmarking process provides the "what" and "why" for a process benchmarking study.

Although there are as many different benchmarking types, processes, and methods as there are companies, each fundamentally follows the same process and steps originally introduced by Xerox in the 1980s. Xerox coined the concept and definition of benchmarking at the turn of the 1970s and 1980s.[109] Robert Camp—while working at Xerox—developed ten benchmarking process steps that he organized into four phases: planning, analysis, integration, and action (see Figure 9). He emphasized the importance of the first three steps: identifying what to benchmark, who to benchmark, and where to acquire the information.[110]

109 Owen and Kheiner 2015, 21–22
110 Camp 1993, 25–26

PLANNING - PHASE 1

1. Identify what is to be benchmarked
2. Identify comparative companies
3. Determine data collection method and collect data

ANALYSIS - PHASE 2

4. Determine current performance gap
5. Project future performance levels

INTEGRATION - PHASE 3

6. Communicate benchmarking findings and gain acceptance
7. Establish functional goals

ACTION - PHASE 4

8. Develop action plans
9. Implement scecific actions and monitor progress
10. Recalibrate benchmarks

FIGURE 9

Robert Camp's original ten benchmarking process steps. Adapted from Camp's work (1993).

As it concerns design, benchmarking is the method most used by designers for gathering data, insights, inspiration, and ideas. There are numerous platforms explicitly designed for searching design inspirations, such as Pinterest[i], Dribbble[ii], and Awwwards[iii]. In addition to technical aspects, benchmarking helps to piece together mood boards, inspires the draft of the visual appearance of a product, and provides an overall picture of what already exists. Aside from UX research, where benchmarking has gained a foothold,[111–112] benchmarking methods and processes in the design industry need further research.

The benchmarking process of this thesis was based on two concepts that examined the characteristics that helped form sophisticated web-based products. Both concepts informed and guided the benchmarking process by providing insights on what to benchmark when the aim was to design and build sophisticated, modular, and scalable web-based products. Moreover, these two selected concepts did not provide two different approaches; they complemented each other and formed a unified and extensive idea of the goals for the benchmarking process of this thesis.

The first concept was based on a study by Sung-Eon Kim, Thomas Shawn, and Helmut Schneider. They coined six central criteria to evaluate websites:[113]

1. **Business function**
   Questions related to products or service information, orders, and transactions within an e-commerce platform.

i www.pinterest.com
ii www.dribbble.com
iii www.awwwards.com

111 Allen and Chudley 2012
112 Interaction Design Foundation n.d.
113 Kim, Shawn and Schneider 2003, 20

2. **Corporation credibility**
   Measures the identity of the corporation behind the website and its purpose of business.

3. **Contents reliability**
   Measures how trustworthy the material is on the website.

4. **Website attractiveness**
   Measures how visually appealing the site is. An attractive website draws visitors to revisit the site.

5. **Systematic structure**
   Measures how consistent the site's information architecture is and how recognizable components are.

6. **Navigation**
   Investigates the ease of navigation between the site's pages or views.

The second concept is coined and introduced by Giorgio Brajnik. He states that the quality of a website is formed by a vast number of factors that he categorizes into three primary categories based on their types: task-related, performance-related, and development-related factors (see Figure 10). He additionally claims that "quality models used in software engineering can be applied to website engineering and that design guidelines and usability evaluation techniques and tools are powerful ingredients of quality models." [114]

114 Brajnik 2001

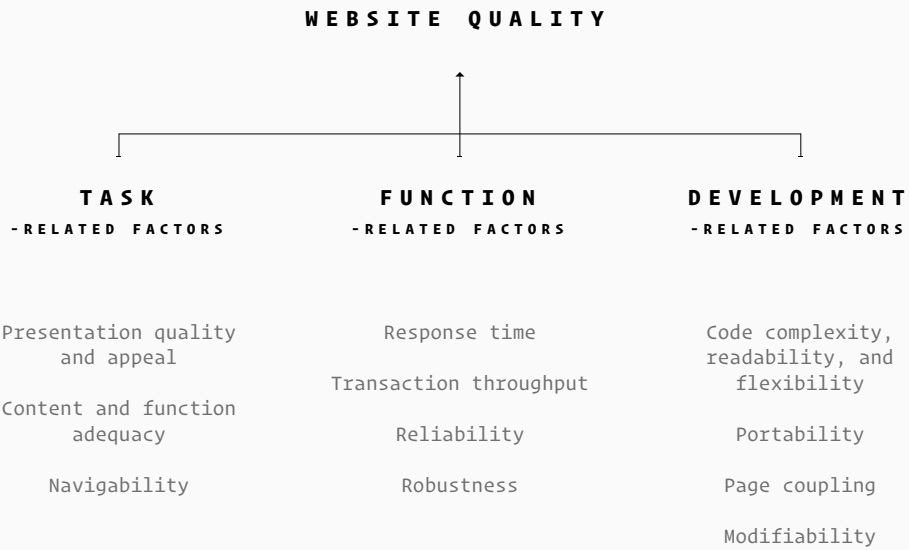


FIGURE 10

Three primary types of categories to define the quality of a website. Adapted from Giorgio Brajnik's work (2001).

Similar to other research methods, benchmarking has advantages and disadvantages. Benchmarking can reveal valuable insights and best practices at its best. Benchmarking can additionally be misleading since the circumstances under which competitors achieve their market position and industry expertise cannot be measured. For example, the reasoning behind architectural choices or understanding competitors' long-term goals cannot be inspected, and the acquired data can be unsuitable for adaptation in the rare case where competitors' long-term goals are inspectable.

Benchmarking, as a data collection method, has been criticized for being a time-consuming process that may not provide any useful information. Moreover, benchmarking is seen as a process to catch up with competitors rather than as a process to achieve market dominance.[115] In terms of data collection, planning and conducting interviews takes time, and interviewees may be hesitant to reveal sensitive issues. The response rate of surveys are generally low, and survey results rarely generate creative ideas.[116]

When it comes to design, there is a line between benchmarking and mimicking. Designers do not see benchmarking as a data comparison tool but rather as a method to collect design inspiration. Particularly in the design industry, benchmarking can foster mediocrity since it does not generate new ideas since it reveals at a glance at what has already been made and published. Hence, it should not lead the design process—notably when the idea is to produce an original piece of design work.

115 Morreale and Terplan 2010, 125–26
116 Watson 2008, 87–88

CHAPTER V

Research Process

This chapter outlines the research process of this thesis. It starts with an introduction and reveals additional background information regarding the project. The research process was formed from two main phases—benchmarking and coding.

The research project of this thesis was formed from two phases. The first phase was a benchmarking process, during which the data was collected for the second phase, a data coding and analysis process. The first phase constituted a foundation for this thesis since the second phase was entirely based on the findings of the benchmarking process.

This thesis selected three web-based streaming services as its primary source of insights, data, and inspiration. The selected services were Deezer, Tidal, and SoundCloud. However, only their web-based platforms were included in this study for two main reasons. Firstly, the focus of this thesis lies in sophisticated and modular web-based products that additionally utilize design systems. Secondly, native mobile apps are not as comparable and inspectable as web-based products are in this context.

Deezer

Deezer—originally Blogmusik.net—is a French online music streaming service founded by Daniel Marhely and Jonathan Benassaya in 2006. The company's original idea was to provide free and legal music as a counter to pirated and illegal music. The service relied on advertising, both banner and voice ads, during its early years. However, they introduced their first paid subscription, Deezer HQ (4.99 euros a month), in 2009. Moreover, their breakthrough was the launch of their first iOS app—along with the new Deezer Premium subscription (9.99 euros a month) that provided access to the app.[117]

Deezer was selected because it has been a web-first service and platform from its beginning. It is additionally an ideal example of a robust web-based platform that follows the latest tech trends and provides consistent user experience between its numerous touchpoints.

117 Rozat 2011

Tidal

Tidal is based on WiMP, a music streaming service and platform developed by Aspiro and acquired by Project Panther Bidco (owned by Shawn Carter or Jay-Z). It offers high-fidelity (hi-fi, hifi, or hi-res) music and video streaming. The service was republished and branded as Tidal in 2015.[118] Currently, Tidal, as well as Deezer, is a subscription-based streaming service that has been heavily branded and marketed as an artist-friendly and artist-owned service. Tidal claims to provide the highest royalty rates, implying that artists—and entire production teams—can receive higher revenue than they would typically get from competing services such as Deezer and Spotify.[119]

Tidal was selected for this research because it is not as established as, for example, Spotify and Apple Music. Moreover, Tidal's web platform is continually developing, and the service is finding its position in the markets.

SoundCloud

SoundCloud is a Berlin-based music distribution and sharing platform founded by Alexander Ljung and Eric Wahlforss in 2007. Its original intention was to provide an online platform for musicians to host and share their recordings and receive feedback from users. However, since then, the service has grown to be one of the leading online music services. SoundCloud's breakthrough feature was its scalability—today, SoundCloud offers various choices (iframes, widgets, plugins) to embed their music player to external websites and blogs.[120] Hence, SoundCloud can be seen as "YouTube for indie music." Nevertheless, despite the

118 Sisario 2015
119 Yoo 2015
120 Buskirk 2009

platform having 25 million users and over 200 million records,[121] it faces difficulty in monetizing its service and securing stable income revenues—nearly closing the service in 2017.[122]

The platform was included in this study to bring variation since Deezer and Tidal are nearly identical services. Furthermore, despite the selected streaming services appearing similar, they are different under the surface. For instance, Deezer is a sophisticated platform, whereas Tidal is an underdog among streaming services.

5.1

Benchmarking Process

The benchmarking process consisted of three main steps: (1) preparing the benchmarking process, (2) gathering data and insights, and (3) comparing the gathered data and insights. The primary aim of the benchmarking process was to collect data for the second research phase and provide a comprehensive understanding of how the selected streaming service platforms were designed and developed. The benchmarking process was conducted by using Google's Chrome web browser—mainly its development tools (DevTools)—and GitHub's Atom text editor. Additionally, various add-ons and plugins were used to structure, analyze, and import and export data between the platforms.

Step 1: Preparing the Benchmarking Process

The first step, preparing the benchmarking process, consisted of various tasks related to planning and preparation. Screenshots were taken, and the source codes and other files of the selected streaming services copied

121 Lunden, 2020
122 Deahl and Newton 2017

as a precaution against drastic changes occurring during the research process.

A Chrome add-on called Full Page Screen Capture[i] was used to obtain fullscreen screenshots. The add-on captured entire pages and saved them as .png or .pdf files. However, it did not capture moving or appearing elements, such as fixed navigation bars or popup images. In this case, however, it did not matter since interactions, animations, and fixed elements were not under examination at this stage of the research process. After taking the screenshots, several parts of the source codes were exported and saved as Sass files by using the Atom text editor. However, the original codes were shown as minified CSS files. Therefore, in order to convert the files back to Sass (the original files were either Sass or Less files), the contents of the original files were copied to an online converter called css2sass[ii] (that converts CSS to Sass) and then back to Atom. The code files were additionally beautified by using an Atom package (plugin) called Atom-Beautify[iii].

Step 2: Gathering Data and Insights

The second step of the benchmarking process began with the creation of low-fidelity (lo-fi, lo-res) wireframes while simultaneously writing memos. The layouts of the streaming services were converted into lo-fi wireframes in order to find repeating design patterns, scrutinize their architectural structures, and understand naming conventions and best practices behind the platforms. The wireframes were created, at a minimum, from three main views: homepages, dashboards (after the user had logged in), and album or playlist page (see Image 2 and Appendix).

i www.gofullpage.com
ii css2sass.herokuapp.com
iii www.atom.io/packages/atom-beautify

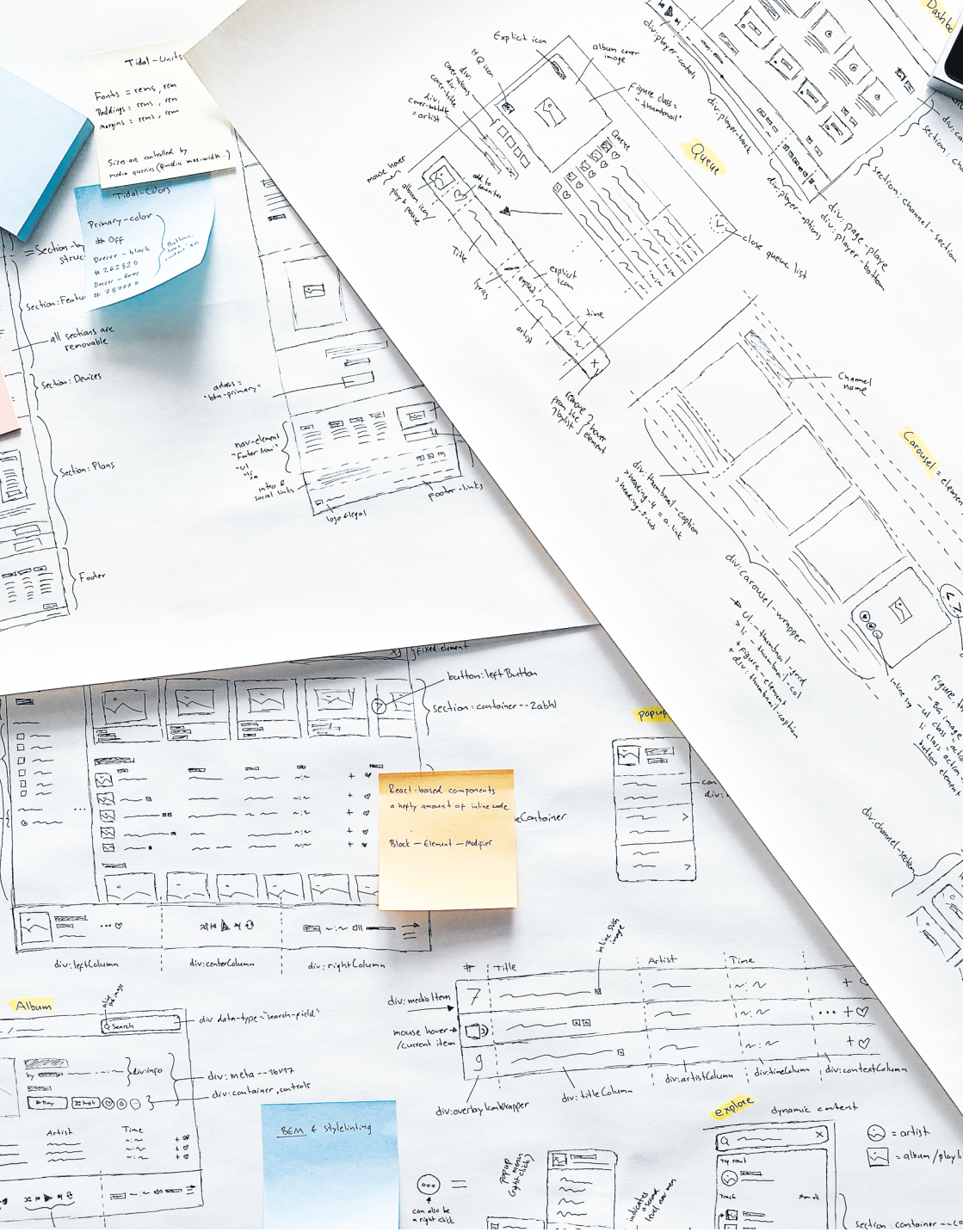


IMAGE 2

Low-fidelity wireframes of the streaming services.

The benchmarking process additionally observed individual elements, such as queue lists, popover menus, sidebars, and list items. The wireframes were created with black markers and a roll of sketch paper. The memos were written on paper and post-it notes—several of which were attached on top of the wireframes. Structural elements within the wireframes were identified and named after their structural HTML element types (div, figure, section, nav, button, article, and footer), and CSS class names.

Step 3: Comparing Data and Insights

The last step of the benchmarking process was to systematically compare the selected platforms. The following criteria were benchmarked from a design perspective: usability, responsivity, flexibility, applied front-end and UI frameworks, and visual languages. Numerous design patterns were additionally benchmarked and inspected in order to build an understanding of how the platforms worked—for instance, how to search for a specific recording or how to create a new account. In order to understand how the platforms were created, an online-based service called BuiltWith[i] was used. BuiltWith is an Australian web-based service that provides information on how a website has been built. The service summarized and categorized all technologies, libraries, and frameworks that the websites used.

The responsiveness and adaptability aspects were manually tested by simulating the platforms through Chrome's Device Mode function.[ii] Adaptability was benchmarked by altering the structures of the platforms' HTML elements and modifying their document object models (DOMs). Furthermore, the structure of the layer hierarchy was benchmarked by modifying CSS values (see Image 3).

i www.builtwith.com
ii developers.google.com/web/tools/chrome-devtools/device-mode

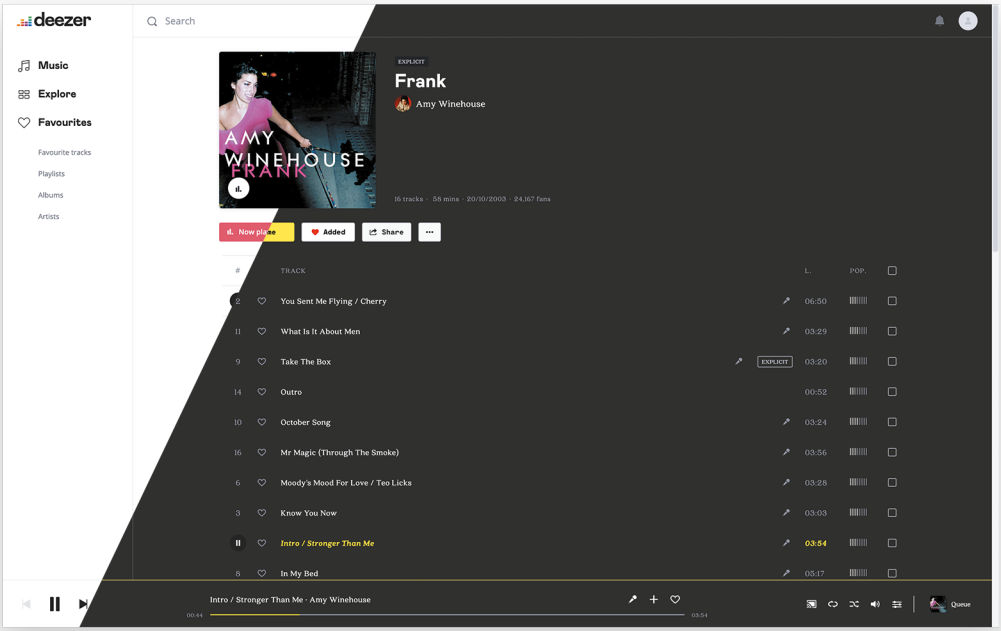


IMAGE 3

Deezer's album view. The light UI version was original, while the dark UI version was altered through modification of the original CSS properties. The altered, skinned version was made to reflect the visual design of this thesis.

5.2

Data Coding and Analysis Process

The first phase of this research provided us with insights, materials, and data needed to start the second phase: the data coding and analysis process. The data coding and analysis process consisted of three steps and coding types: open, axial, and selective coding.

Step 1: Open Coding

The data coding process began by printing both CSS files and HTML structures of the selected platforms. However, in order to keep the data coding process as systematic and manageable as possible, only the

benchmarked elements and pages were scrutinized (see Appendix). The open coding step was formed from three levels by using three coding approaches that are listed in Section 4.1: word-by-word, line-by-line, and incident-by-incident approach.

The findings discovered during the open coding stage were categorized according to their content types. For example, website layouts were dismantled to their HTML element types—returned to divs, sections, buttons, headers, articles, and list and navigation items. These categories were further scrutinized during the second step of the data coding and analysis process.

Step 2: Axial Coding

The second step of the data coding and analysis process was axial coding (see Section 4.1). Axial coding analyzes connections between categories and their subcategories. The axial analysis stage was conducted similarly to the benchmarking process, with the use of black markers, post-it notes, and a roll of sketching paper (see Image 4).

Step 3: Selective Coding

The selective coding step finalized the data coding and analysis process. It further scrutinized mid-level categories and found a set of high-level categories of modular design architecture by searching the common denominators amidst the mid-level categories. As discussed in Section 4.1, the goal of selective coding is to find and form a core category, an umbrella term that covers all other categories discovered from the acquired data. In this case, the umbrella category was labeled “architectural dependency.”

IMAGE 4

The open coding process was conducted using paper and writing materials. The process had three phases, and each phase was built on the previous phase.

CHAPTER VI

Research Findings and Insights

This chapter reports the research findings and insights acquired during the research process.

The benchmarking process revealed both design and architectural structures behind the selected streaming services. The central intention behind the benchmarking process was to investigate and illustrate how the discovered design and architectural structures were designed and implemented by creating wireframes of the selected streaming services. The research findings demonstrated how one could build and design various sophisticated web-based products. The same visual language could be achieved with various underlying architectures. The selected streaming services, and Deezer and Tidal, in particular, shared similar visual characteristics and design patterns; however, the products were constructed differently.

6.1

Findings of the Benchmarking Process

The benchmarking process focused on creating informative wireframes of the selected web-based streaming services. These wireframes illustrated architectural elements of the streaming services (HTML structure elements) and the frontend coding structures of these discovered architectural elements (CSS classes).

A wireframe does not typically include information about how something has been or will be built since wireframes are generally sketched before the programming process begins. However, it seems to be beneficial if different sections and elements within a wireframe are separated and clearly indicated—particularly for web developers who implement the desired visual language into the web-based product. Wireframes that have the coding-side of the design ideated can guide designers in creating realistic mockups. Pixel-perfect layouts are difficult to create since no design software is currently capable of creating

genuinely responsive design elements. Hence, responsivity aspects are not—and cannot be—significantly considered during design processes. For example, when creating separate variations of the same layout for mobile, tablet, and computer screens based on their typical screen sizes, some screen sizes are inevitably excluded.

The crafted wireframes were readily comparable with each other since they shared similar structural elements and were otherwise nearly superficially identical. However, the elements were dissimilar under the surface, each having a different design architecture. Despite this information being more relevant to frontend developers than designers, designers can additionally apply it. For example, if it is known that a web-based product is to be based on Bootstrap, designers can efficiently start their design processes by utilizing and skinning Bootstrap's UI components. Skinning, in this context, means changing only the visual attributes of a component, such as colors, fonts, and paddings based on the branding of the product (see Image 3). The functionality of the components remains untouched. Additionally, designers can use ready-made UI kits instead of designing from the ground up.

6.2
Findings of the Data Coding and Analysis Process

The data coding and analysis process consisted of three steps: (1) an open coding step that was conducted by applying three data coding approaches, (2) an axial coding step during which the categories were established based on the discoveries of the first step, and (3) a selective coding step that concluded the data coding and analysis process (see Image 5).

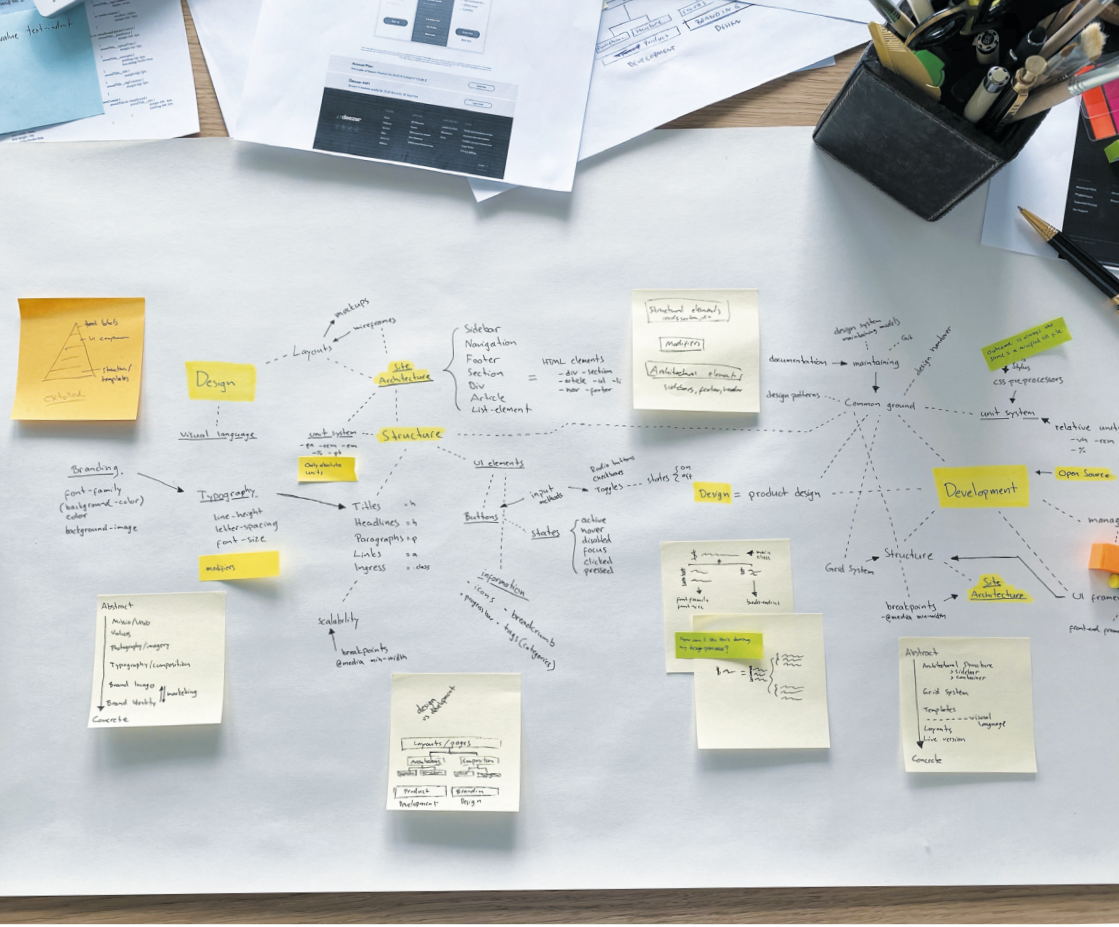


IMAGE 5

An overview of the data coding process.

Step 1: Open coding

The word-by-word data coding approach sifted through source codes at a detailed level. This approach provided foundational information, such as which unit types, color values, and HTML element types were used to create the streaming services. Although this data may not be useful outside of its original context, a few discoveries can still be found. For instance, unit types used in websites are generally not the same as those

used in design applications. Relative length units, such as rem, vh, em, and %, cannot be used in common design applications. Vice versa, unit types used in design applications are generally only absolute (pixels, points, mm, in) and, thus, not intended to be used on the Web. Therefore, one can argue that creating pixel-perfect layouts is rarely feasible since the outcomes—the coded versions of the same layouts—may rarely be identical to the original layouts, only corresponding at their best. However, in this research, one common denominator was found between units used in design applications and units applied in web development: pixels. For example, both Deezer and SoundCloud applied pixel-based typography and layouts, whereas Tidal's typography and layouts were based on rem values.

The line-by-line approach revealed how the services were built with more detail than the word-by-word approach. For instance, external libraries, website analytic tools, search engine optimization (SEO) scripts, and services offered by third parties were assessable. Additionally, some internal communication and documentation were visible. For example, Tidal's source code revealed that they were using Zeplin before moving to Figma, and SoundCloud's source code had several tasks from their internal to-do list.

The incident-by-incident approach looked at the larger picture and unveiled the architectural structures of the coding-side of the streaming platforms. It was discovered that all selected streaming services followed systematic and self-explanatory naming conventions, but Deezer's structure was more sophisticated than the other two. The approach underlined how a consistent naming convention plays a leading role in both design and web development. For instance, Sketch builds component libraries according to component names (see Image 6). Inconsistent naming conventions can create confusion—particularly when platforms grow and become increasingly complex (see Section 3.4). One of the insights achieved

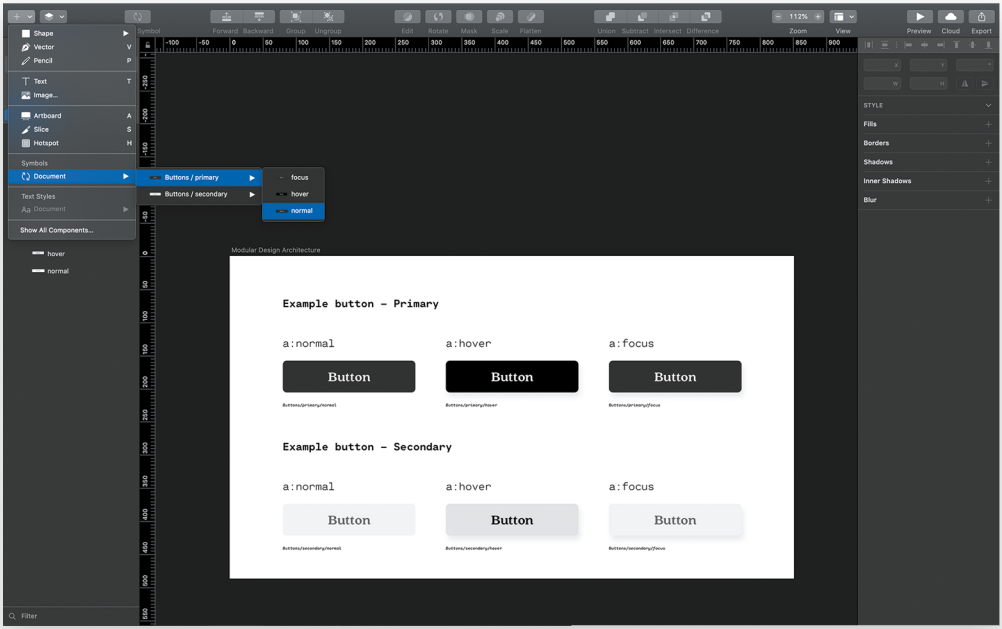


IMAGE 6

Sketch builds component libraries according to component names.

during the open coding process was how a coherent—or identical—naming convention could be applied in both design and web development.

The incident-by-incident approach exposed how terminology used in the design industry—for example, in the atomic design approach—and terminology used in web development collided with each other. There were no molecules or organisms per se, only sections, articles, and divs. Therefore, precisely naming design elements, for example, by using the Block Element Modifier (BEM) and Scalable and Modular Architecture for CSS (SMACSS) methodologies, could enhance understanding between designers and developers.

Step 2: Axial Coding

After the first data coding step loosely processed the data acquired from the benchmarking process, the axial coding process studied this data further. The axial coding process classified this unstructured data as the sub-level settings of modular design architecture that formed foundations for visual elements. Concretely, these sub-level settings were CSS declarations that formed declaration blocks (see Section 3.4). In contrast, these sub-level settings were object properties or layer, font, and paragraph styles in design applications. These foundations of visual elements were formed from two categories that were labeled as definitions and modifiers. The definitions category consisted of properties that defined something. In contrast, the modifiers category held values that defined appearance. For instance, in CSS, the font-family value defined the font (what the font is), whereas the color value defined its color (what the font looks like).

The second step of the data coding process discovered a new level of modular design architecture: combinations and compositions. This level was observed from two perspectives: design and development. Concretely this level forms declaration blocks (development) and visual design elements (design). These visual design elements are known by various names, although they have the same meaning. For instance, in Adobe's Photoshop and Illustrator, these visual design elements are known as grouped layers whereas in Adobe XD and InVision Studio they are known as symbols.

The axial coding phase additionally discovered another level of modular design architecture (mid-level), and it was labeled as the categories within modular design architecture. These categories were observed from both design and development perspectives. However, only a few of these categories emerged from both perspectives.

Scalability, structure, common ground, and unit system were categories that concerned both perspectives. More precisely, the common ground category included shared concepts and properties, such as design handovers and documentation, that were the same both for designers and developers.

The discovered relationships between these categories were marked in two ways (see Image 7). A dashed line indicated that the relationship affected both categories—when one changed, the other changed as well. In contrast, a solid line with an arrowhead indicated that a specific category only affected the other one(s). For instance, layouts are based on wireframes and mockups are based on layouts. In other words, wireframes should be created before forming layouts and mockups created after the layouts have been defined. Several of these categories were found in both design and development perspectives, such as the unit system category. However, the unit system category was not under the common ground category since its content was not the same. Designers only used absolute units, while developers used both relative and absolute units.

Step 3: Selective Coding

The final data coding step concluded the findings and insights established during the first two steps. The outcome of the selective coding process was a concept of a new framework, which addressed the degree of architectural dependency of the inspected component, module, or system within a specific modular design architecture. This framework was established during the selective coding step by combining and collating memos that were written throughout the research process (see Image 8).

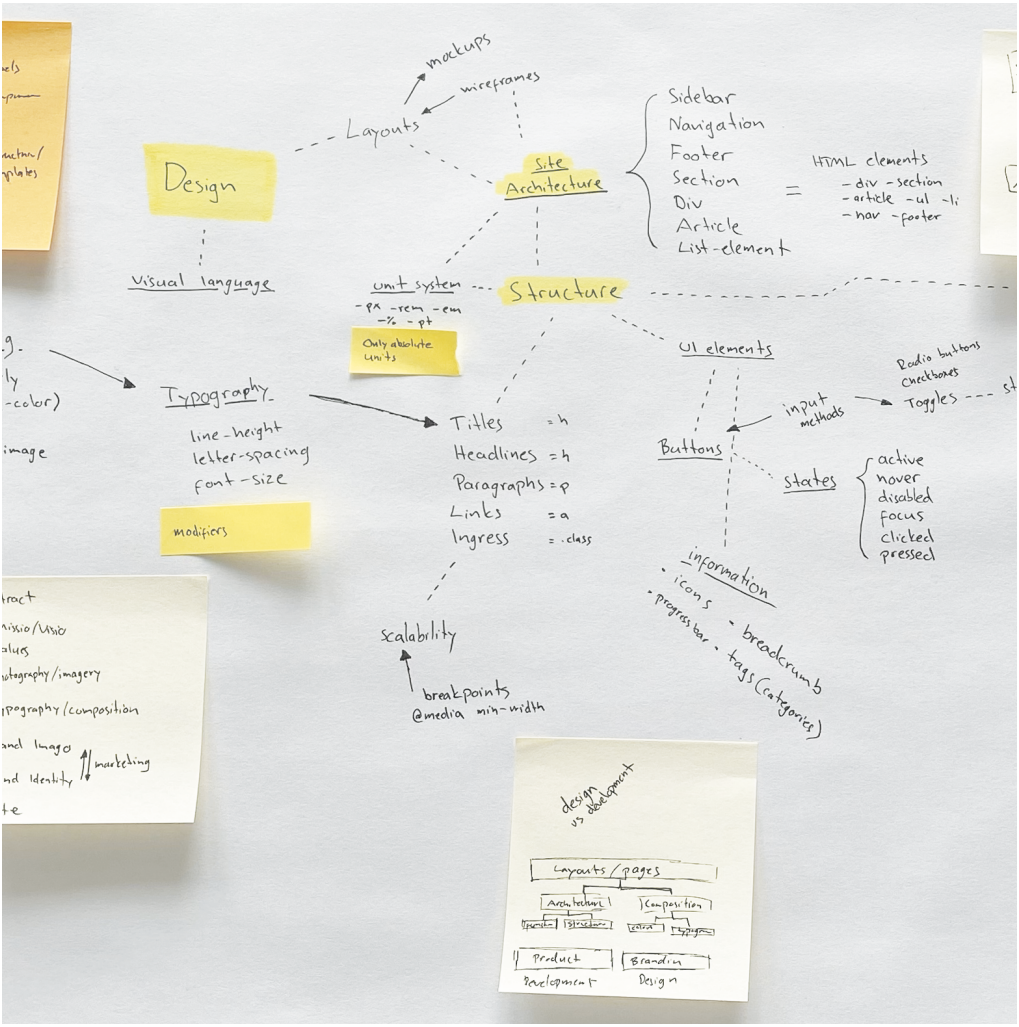


IMAGE 7

The axial coding phase scrutinized the relationships between the categories established in the previous stage.

IMAGE 8

A collage of notes, source codes, and layouts that guided the research process. The materials were mainly created during the coding phase.

6.3

The Framework of Architectural Dependencies

The research process of this thesis discovered a new approach to defining and inspecting modular design architectures of web-based products. This coined framework can aid both designers and developers prioritize design tasks based on importance, structural characteristics, and post-editing capabilities in order to build scalable and modular design architectures for web-based products.

The degree of architectural dependency is a property that indicates how essential a particular definition, category, component, module, or system is within the product's design architecture. The research found that the higher the degree of architectural dependency, the more challenging the definition, category, component, module, or system was to modify after the development process had begun. For instance, a selected naming convention had a high degree of architectural dependency, implying that changing it after development had started would require significant work. In contrast, definitions that had a low degree of architectural dependency, such as a selected primary font, could be effortlessly modified. For instance, changing the definition of the primary font from Open Sans to Noto Sans could be done within a minute—but only if the product were built upon modular design architecture.

The framework presents and organizes an internal hierarchy within a modular design architecture, and is applicable for both designers and developers. For instance, the framework can guide designers in creating new coherent visual assets and maintaining the desired consistent visual language. On the other hand, frontend developers can apply the framework to fostering the modularity and scalability of the design system. The framework can additionally help other stakeholders

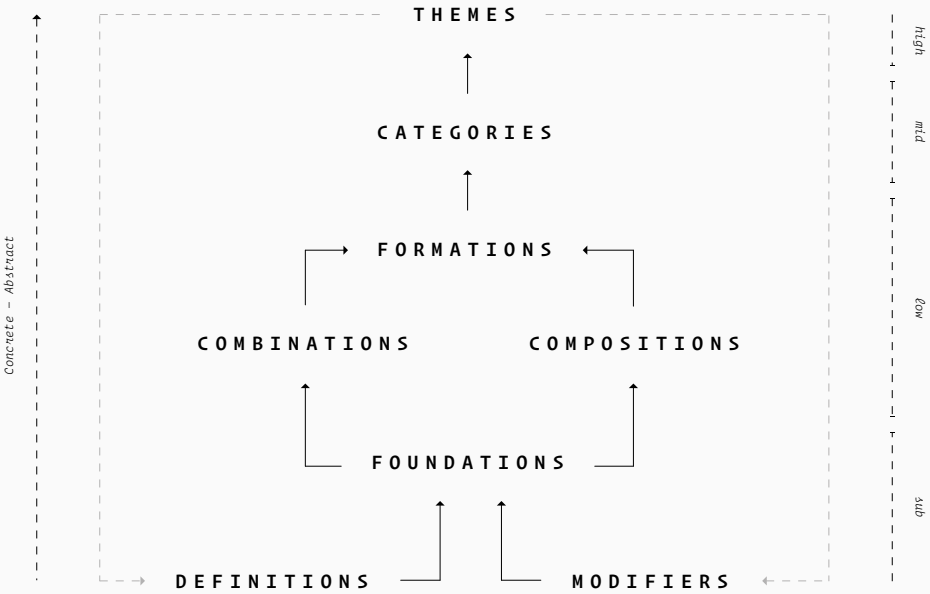


FIGURE 11

The research findings discovered four levels of modular design architecture: foundations, formations, categories, and themes. These levels had a certain degree of architectural dependency, shown on the right side of the figure. Arrows indicate interdependencies amidst the levels and underline the bottom-up structure of modular design architecture.

more thoroughly understand the product since the framework concretely illustrates the structure of the product.

The research process investigated the interaction between different categories found in modular design architecture. These interactions had different levels of architectural dependencies amidst them, and they were generally unidirectional—lower level affected the higher level but not vice versa (see Image 7). The sub and lower levels were examined during the first step of the data coding process, and it was found that the sub-level created foundations for mid-level formations. Nevertheless, both levels were excluded from the axial coding process because they were product-specific definitions rather than being generalizable. For example, a color theme and typography, among other visual aspects, were frequently based on the brand identity of the product or personal preference. The selected streaming services had different and incomparable design architectures when examined at their lower levels. Therefore, it was not feasible to focus on these levels since they were not generalizable.

A few mid and high-level categories and themes were found through the data analysis (see Image 8 and Figure 8). Both levels were looked at from both design and development perspectives. For instance, the following categories and themes were found:

Categories: branding, typography, UI elements, layouts, unit systems, UI frameworks, product management, product design, grid system, documentation, structure, scalability, design handover, and design patterns.

Themes: site architecture, visual language, common ground, and product maintenance.

Essentially, the established categories formed the discovered themes. These categories and themes were highly generalizable, and it is recommended all sophisticated web-based products have them. However, the amount and names of these categories and themes can vary. They were, particularly in this study, grounded to their context, and each web-based product could have its own, unique modular design architecture.

6.4
Research Insights

The grounded theory approach proved to be an ideal method for scrutinizing abstract themes, such as design processes, design architecture, and modularity. Although the findings of the research process developed a theoretical framework—the concept of architectural dependency within a modular design architecture—the framework can be readily put into practice. The following three insights were established during the benchmarking and data analysis processes to address the research questions of this thesis:

Insight 1: Do Not Build on Sand

By starting a design process from a high level of architectural dependency, a robust foundation is built for the product. The foundation of a design system—as well as the foundation of modular design architecture—should be carefully defined both technically and structurally before development begins since these definitions cannot be readily modified later.

Hence, to kickstart the product development process, it may be feasible to start the product design process by using a UI kit and UI framework before proceeding to custom elements after a minimum viable product (MVP) version has been created. Although they may limit

the design options, UI frameworks such as Bootstrap and Semantic UI already contain robust and modular foundations for products.

Insight 2: Lo-fi Over Hi-fi

In order to accelerate product design processes, designers can create lo-fi elements instead of hi-fi mockups. Production-wise, there are no reasons to create pixel-perfect mockups of various scenarios since outcomes are rarely the same as the mockups.

After the overall structure of the website is defined, lo-fi mockups provide sufficient information to start the development process. However, hi-fi mockups can be created, for example, for marketing purposes after the development process has started since visual properties generally have a low degree of architectural dependency; hence, they can be readily modified at any time. Moreover, creating a hi-fi mockup requires a high time commitment, and yet, the newly created mockup can become obsolete at the moment it is handed over if its elements have changed during or after its creation.

Insight 3: From Abstract to Concrete

The building process of modular design architecture can begin as soon as the production team has defined basic rules, settings, and guidelines. Starting the definition process from top to bottom (see Figure 11)—from categories and themes that have a higher degree of architectural dependency—would provide a straightforward process for establishing modular design architecture. However, this top to bottom approach is only feasible when defining architectural aspects, not when designing or developing UI assets.

For example, as discussed in Section 3.4, when defining the visual language of a product, the process should begin from the foundation (see

Figure 5 in Section 3.4) instead of starting to draft layouts from scratch. Production-wise, these are minor details; focusing on letter-spacing does not make a MVP more established, but only makes it moderately more refined. As previously stated, visual elements generally have a low degree of architectural dependency, and hence, they can be modified at any time.

CHAPTER VII

Discussion

This chapter presents the discussion section of this thesis. It additionally provides a summary of the obtained insights.

One could say that the design industry has been late in adapting and learning from other disciplines. As discussed in Section 3.1, many modern concepts and best practices that are advantageous in the design industry were initially developed in the late 1970s. For instance, Christopher Alexander introduced the concept of design patterns in the late 1970s,[123] and Edsger Dijkstra presented the idea of a component-based design approach in the early 1970s.[124] However, it took approximately four decades before the design industry caught up with these concepts (among others) and adapted best practices, theories, and frameworks from other disciplines.

Who Sets the Rules?

The bond between the design and software industry is insufficiently standardized, as numerous challenges need to be tackled before they can work together along the production line. In contrast, the bond between the design and print industry is inseparable and highly standardized. For instance, all commercial printers have been calibrated to support specific and ISO-standardized color profiles, such as Coated Fogra 39 when printing on coated paper and Uncoated Fogra 29 for uncoated paper. The printing house sets boundaries and design rules for designers; designers have to follow these rules or printed materials may contain blurry images and unsharp texts. On the other hand, when designers follow the requirements set by printing houses, the outcomes are pixel-perfect, sharp, and highly predictable.

When it comes to designing for digital platforms, there are no precise design requirements set by the software industry; only vague guidelines exist. The benchmarking phase of this thesis demonstrated how

123 Alexander, Ishikawa and Silverstein 1977
124 Dijkstra 1972

unique teams could build similar products. Although there were common characteristics among benchmarked web-based products—such as site structures, design patterns, and used iconography—under the surface, they significantly differed.

One could say that the software industry could become the new print industry. As discussed in Chapter 3, many of today's leading design trends follow the best practices and theories found in the software industry, such as pattern and component libraries, the modular product design approach, and consistent naming conventions. As long as designers produce materials for production purposes—whether the outcome is a flyer or a website—they should create design assets according to the standards set by the selected platforms. All designed materials, both physical and digital, are similarly bound to their platforms, similar to how an oil painting is bound to the canvas on which the artist has painted it.

The data coding and analysis phase of the research process revealed the importance of the common language among stakeholders in understanding a product and its structure. The research findings showed how essential a consistent and self-explanatory naming convention was in order to build robust products. It was additionally interesting to discover that the more sophisticated the naming convention of a streaming service, the more robust its platform was. Of note, the research project was limited to three similar platforms; thus, no generally applicable conclusions could be drawn from this insight. However, since product design processes are not linear, one-off tasks but rather continuous dialogues between designers and developers, it is crucial to establish a common ground for all stakeholders. A consistent naming convention both in design and development seems to play an essential role in establishing this common ground.

Since the software industry creates the canvas on which the designed web-based product is to be built, it makes sense for the software

industry to additionally set rules for designers and guide them through the product design process. The resulting outcome would be a digital product that mimics designed layouts (see Section 5.2). Moreover, there is a frequent risk that the designed layouts are not suitable for production if the platform-imposed constraints are not identified at the beginning of the design process.

Where Are We Now?

This thesis process provided numerous insights into the design and software industries. It was noticeable during the research process that the design industry is evolving and increasingly adapting elements from the software industry. Additionally, the literature review of this thesis supported this observation. For instance, the current professional literature regarding design systems is profoundly influenced by the software industry.[125–126]

The literature review, data coding phase and analysis phase revealed that only a few design approaches or best practices are available for designing sophisticated web-based products. Approximately all publications were written for frontend developers rather than for designers. One of the most frequently cited books regarding design systems was Brad Frost's *Atomic Design*.[127] However, the book was an adapted component-based product development approach from the software industry that had been tailored to suit the needs of the design industry.

There were many design applications available that were specifically focused on design handovers from designers to developers

125 Curtis 2010
126 Godbolt 2016
127 Frost 2016

or the creation of UI assets, such as Sketch, Adobe XD, Figma, Zeplin, InVision, and Framer (framer.com). The question was which one(s) to use and why. Presently, the choice seems to be made based on designers' preferences rather than the requirements set by the selected platforms. For example, Tidal changed from Zeplin to Figma, and Deezer used both Sketch and Zeplin. However, at the time of writing, Sketch is more advanced in crafting design systems than any of the previously mentioned design software.

The Future Lies in Software Development

Design has become the foundation of modern and robust businesses, mainly due to the rise of the startup era.[128] As digital services, platforms, and applications conquer the world, there may be an increasing demand for attractive digital interfaces to compete for customers' attention. Currently, the importance of great design seems to be finally understood and valued amidst all stakeholders within a company.[129]

The topic of this thesis, modular design architecture, decreases barriers between designers and developers. It focuses on the commonalities between the design and software industries and guides both to complement each other. One could say that without ideal design, there is no ideal product and vice versa; great visuals do not help if the software does not function properly. Moreover, modular design architecture provides an opportunity for both developers and designers to speak up early.

The established framework that addresses architectural dependencies within modular design architecture (see Figure 11 in Section 6.3) offers insights into which areas need to be settled before continuing the product design processes. As Section 2.2 revealed, modularity can

128 Chandler 2018
129 Sheppard et al. 2018

foster scalability and rapid product development, allowing a non-linear and agile way of working. The beginning of the third chapter follows this conversation by summarizing how a linear way of working is not a feasible approach to building sophisticated web-based products. Moreover, the established framework implies that not all modules need to be defined before the development process can begin. Only major structural guidelines, such as breakpoints of the layouts and design rules, must be determined in advance.

The research process revealed how modular design architecture emphasizes the shared processes between production team members at the beginning of the product's life cycle. For instance, different levels of modular design architecture—foundations, formations, categories, and themes—are the same for both designers and developers, as shown in Figure 11. Therefore, this underlines that designers and developers should work seamlessly together from the beginning. The process of defining modular design architecture and establishing a design system for a product is the same for both parties. All unaddressed inconsistencies between the two sides of the same UI, the designed and coded sides, foster incoherency and over time create a UI legacy that can practically impossible to fix later.

In the future, the design industry may be as inseparable from the software industry as it currently is from the print industry. However, the journey would be long as many improvements would need to be made beforehand. For example, no current design application supports genuine modularity or design elements that are dynamic, responsive, or code-based.

CHAPTER VIII

Conclusions

This chapter concludes this thesis and reflects on its research findings and relevancy. It additionally suggests future development ideas.

This thesis aimed to identify effective strategies for defining modular design architecture for web-based products. Based on a qualitative analysis of benchmarked web-based streaming services, it can be concluded that modularity, together with a consistent naming convention and a systematic design approach, provides a well-established product environment for accelerating product design and development processes. Hence, these are essential factors to consider when defining modular design architecture at the beginning of the product's life cycle.

The research findings indicate that potential mistakes and misdefinitions at the early stage of product design can create an inconsistent design legacy that is difficult and costly to fix. By analyzing repeating design patterns, product structures, and source codes of selected web-based streaming platforms, this thesis has addressed how uniquely different teams can define and build their products while managing to contain a high degree of modularity and product flexibility.

Answers to the Research Questions

Benchmarking and the methods found in grounded theory methodology proved to be useful approaches for collecting, categorizing, and analyzing data when the examined themes were abstract. Additionally, other interdisciplinary methods and theories were applied and adapted to suit the needs of this thesis.

The research findings provided various insights on the research questions. The first research question, "How can modular design architecture accelerate product design processes when creating sophisticated web-based products?" was partially addressed during the literature review, after which the answer was completed by insights and findings gained from the research process. The conclusion is that a thorough modular design architecture enables both designers and

developers to simultaneously work to mitigate potential forthcoming bottlenecks. A robust design architecture constitutes a stable foundation for UI legacy.

The new framework formed during the data coding and analysis process provides a tool for production teams to evaluate and determine which areas need to be addressed before starting the production process. When both designers and developers follow the framework, this can accelerate product design processes from two perspectives. Firstly, when the foundation of the product is well-established, there is no need to redo the same work. Secondly, it helps to prioritize different product design tasks. For example, in order to accelerate product design processes, abstract materials, such as wireframes and the tentative information architecture of the product, are generally more important than polished, pixel-perfect layouts.

The second research question, "How can modular design architecture be defined when designing new web-based products?" was partially addressed during the literature review as well. Its answer is similar to that of the first question. Modular design architecture can be defined by determining the product's design rules, categories, and themes at the beginning of the product design process and prioritizing them according to their degrees of architectural dependencies (see Figure 11 in Section 6.3).

Research Limitations and Validity

As modularity, product design architecture, and design systems remain limited subjects in the academic literature—and particularly, in the field of design—the conclusions, insights, and findings of this thesis are theoretical and bound to their original context. For example, to utilize the established framework as defined in this thesis, the designed product

should be thoroughly modular, standardized, and web-based. Moreover, the production team should consist of both mid to senior-level designers and developers since the management and planning of intricate design entities require relevant design experience.

The topic of this thesis is highly based on best practices found both in the design and software industry and adds limited knowledge to the literature. On the other hand, the concepts and theories introduced in this thesis have already been extensively studied in fields outside of design. However, the presented framework to address architectural dependencies is a theoretical framework that can be additionally applied to scrutinizing modular systems—including those entirely outside of the context of this thesis.

As with any grounded theory study, this thesis has a certain degree of bias in its research findings and results. The research process and its findings were based on the author's views and choices. Therefore, despite an identical research setup, the outcome could differ depending on the researcher. Hence, the result of this thesis can be seen as a conversation starter around the research topic. However, while the selected research methodology limits the generalizability of the results, the research findings provide novel insight into the research topic.

Research Relevancy

The current design literature requires more research to be done before it can be relevant to the present labor market and its constantly changing design trends. There is an extremely limited number of academic papers and peer-reviewed journal articles available that address topics related to the current design trends, such as modular (visual) design approaches, collaboration between designers, and development and management of design systems.

This thesis provides a cross-disciplinary overview of the aforementioned topics. At the moment, designers do not generally understand software development processes or write high-quality and implementable code. Hence, this thesis provides insights from both worlds and covers both product design and product development perspectives, including the creation of modular design platforms and defining their architectural characteristics. The number of similar studies available is limited, underlining the relevancy of this study.

Recommendations and Future Development

The design industry exists in an era of rapid development. Presently, there are no well-established tools available for designers to design and create dynamic layouts and other visual assets: visual design remains static while digital platforms are dynamic. Therefore, further research is required for the transformation from designed UIs to coded UIs.

The research findings demonstrate the fragmentation of the current design industry. For example, as reported in the previous chapter, there are multiple similar design applications available. Sketch is currently the leading design software amidst UI designers and other creatives since it supports additional functionalities through third-party plugins and has the most advanced basic features. However, further advances in technology and design research could provide designers with improved tools and leaner implementation processes between UI design and UI development. For example, a feature or a platform to visually modify React and Vue components without all the current production steps would be useful.

Moreover, future design research around the topic and themes of this thesis could focus on creating frameworks, templates, and more established theories for designers. These could aid and guide designers

and developers—particularly when MVPs are being built from scratch. Moreover, as designers' roles become increasingly influenced by the latest tech trends, there may frequently be room for more advanced design tools, platforms, and methods.

References, Figures, And Images

All references are alphabetically listed based on the Chicago 17th referencing and citation style. The author has created all figures and images unless stated otherwise.

Works Cited

Adobe Corporate Communications. "Flash & The Future of Interactive Content." Adobe Blog. Published July 25, 2017. https://theblog.adobe.com/adobe-flash-update.

Alexander, Christopher, Sara Ishikawa, and Murray Silverstein. *A Pattern Language: Towns, Buildings, Construction*. New York, NY: Oxford University Press, 1977.

Allen, Jesmond, and James Chudley. "Chapter 6: Gaining Useful Insights from Competitor Benchmarking." In *Smashing UX Design: Foundations for Designing Online User Experiences*, 97–104. Chichester, United Kingdom: John Wiley & Sons, Ltd., 2012.

Anand, G., and Rambabu Kodali. "Benchmarking the Benchmarking Models." *Benchmarking: An International Journal* 15, no. 3 (2008): 257–291.

Awwwards. "Web Design Trends 2019: Voice Interfaces, Image Search, Fortnite, Alexa and other crazy things that are rocking our world." Published April 11, 2019. https://www.awwwards.com/web-design-trends-2019.html.

Backlund, Alexander. "The Definition of System." *Kybernetes* 29, no. 4 (2000): 444–451.

Baldwin, Carliss Y., and Kim B. Clark. "Managing in an Age of Modularity." *Harvard Business Review* 75, no. 5 (1997): 84–93.

Baldwin, Carliss Y., and Kim B. Clark. *Design Rules, Vol. 1: The Power of Modularity*. Cambridge, MA: The MIT Press. 2000.

Baldwin, Nate. "Component and Token Naming in Design Systems." Published February 19, 2020. https://medium.com/@NateBaldwin/component-and-token-naming-in-design-systems-366bad54843f

Bogan, Christopher E., and Michael J. English. *Benchmarking for Best Practices: Winning Through Innovative Adaptation*. New York, NY: McGraw-Hill, 1994.

Booz, Edwin, James Allen and Carl Hamilton. *New Product Management for the 1980's*. New York, NY: Booz Allen Hamilton Inc., 1982.

Brajnik, Giorgio. "Towards Valid Quality Models for Websites." In: *7th Conference on Human Factors and the Web*. Published June 2001. http://users.dimi.uniud.it/~giorgio.brajnik/papers/hfweb01.html.

Buskirk, Eliot Van. 2009. "SoundCloud Threatens MySpace as Music Destination for Twitter Era." Wired. Published July 6, 2009. https://www.wired.com/2009/07/soundcloud-threatens-myspace-as-music-destination-for-twitter-era/

Cambridge Dictionary. "Component." Accessed November 23, 2019. https://dictionary.cambridge.org/dictionary/english/component.

Camp, Robert C. *Benchmarking: The Search for Industry Best Practices that Lead to Superior Performance*. Milwaukee, WI.: ASQC Quality Press, 1989.

Camp, Robert C. "A Bible for Benchmarking, By Xerox." *Financial Executive* 9, no. 4 (1993): 23–27.

Cao, Jerry, and Carrie Cousins. *Web Design Trends 2018*. UXPin, 2018. Ebook.

Chacon, Scott, and Ben Straub. *Pro Git*. 2nd ed. Apress, 2014.

Chandler, Clay. "The Meaning of Design Is up for Debate. And That's a Good." *Time* 191, no. 10 (2018): 18.

Charmaz, Kathy. "'Discovering' Chronic Illness: Using Grounded Theory." *Social Science & Medicine* 30, no. 11 (1990): 1161–1172.

Charmaz, Kathy. *Constructing Grounded Theory: A Practical Guide Through Qualitative Analysis*. London, United Kingdom: Sage, 2006.

Chatras, Clément, and Vincent Giard. "Simultaneous Standardisation of Modules and Their Components: A Global Economic Perspective." *International Journal of Production Research* 54, no. 19 (2016): 5722–5741.

Corbin, Juliet, and Anselm Strauss. "Grounded Theory Research: Procedures, Canons, and Evaluative Criteria." *Qualitative Sociology* 13, no. 1 (1990): 3–21.

Cooper, Robert. *Winning at New Products*. Reading, MA: Addison-Wesley Publishing Co, 1986.

Creswell, John W., and Cheryl N. Poth. *Qualitative Inquiry & Research Design: Choosing Among Five Approaches*. 4th ed. International student edition. Los Angeles, CA: SAGE Publications Inc., 2018.

Curtis, Nathan. *Modular Web Design: Creating Reusable Components for User Experience Design and Documentations*. Berkeley, CA: New Riders, 2010.

Curtis, Nathan. "UX Patterns ≠ UI Components." Published January 28, 2017. https://medium.com/eightshapes-llc/patterns-components-2ce778cbe4e8/.

Deahl, Dani, and Casey Newton. "How SoundCloud's Broken Business Model Drove Artist Away." *The Verge*. Published July 21, 2017. https://www.theverge.com/2017/7/21/15999172/soundcloud-business-model-future-spotify-streaming.

Dijkstra, Edsger. "The Humble Programmer." *Communications of the ACM* 15, no. 10 (1972): 859–866.

DiNucci, Darcy. "Fragmented Future." *Print 53, no. 4 (1999)*: 32, 221–222.

Efatmaneshnik, Mahmoud, Shraga Shoval, and Li Qiao. "A Standard Description of the Terms Module and Modularity for Systems Engineering." *IEEE Transactions On Engineering Management* PP, no. 99 (2018): 1–11.

Foote, Steven. *Learning to Program*. Addison-Wesley Professional, 2014.

Frost, Brad. *Atomic Design*. Pittsburgh, PA: Brad Frost, 2016.

Fuller, Matthew. "Software Studies: A Lexicon." Cambridge, MA: The MIT Press, 2008.

Gershenson, John, Gajendra Prasad, and Srikanth Allamneni. "Modular Product Design: A Life-Cycle View." *Journal of Integrated Design and Process Science* 3, no. 4 (1999): 13–26.

Gershenson, John, Gajendra Prasad, and Y. Zhang. "Product Modularity: Definitions and Benefits." *Journal of Engineering Design* 14, no. 3 (2003): 295–313.

Gibbs, Graham J. "Grounded Theory – Core Elements. Part 2." Published June 19, 2010. Video, 6:22. https://youtu.be/dbntk_xeLHA.

GitHub. "*Primer · GitHub*." GitHub. Accessed January 10, 2020. https://github.com/primer/.

Glaser, Barney G., and Anselm L. Strauss. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Chicago, IL: Aldine Publishing Company, 1967.

Godbolt, Micah. *Frontend Architecture for Design Systems: A Modern Blueprint for Scalable and Sustainable Websites*. Sebastopol, CA: O'Reilly Media Inc., 2016.

Gonzalez, Oscar. "Crafting a New Visual Language." *Muzli*. Published August 14, 2017. https://medium.muz.li/crafting-a-new-visual-language-912d3ac8df43.

Google LLC. "Homepage – Material Design." *Material Design*. Accessed January 10, 2020. https://material.io.

Greenberg, Ran. "Naming Convention – 9 Basic Rules For Any Piece of Code." *Wix Engineering*. Published October 10, 2019. https://medium.com/wix-engineering/naming-convention-8-basic-rules-for-any-piece-of-code-c4c5f65b0c09

Hacq, Audrey. "Everything You Need to Know About Design Systems." *UX Collective*. Published May 22, 2018. https://uxdesign.cc/everything-you-need-to-know-about-design-systems-54b109851969

Hauer, Erwin. "Design 2." Vienna, Austria, 1951. Accessed August 6, 2019. http://erwinhauer.com/eh/installations/church-in-liesing-vienna-austria.

Horn, Robert. "Visual Language and Converging Technologies in the Next 10–15 Years (and Beyond)." *National Science Foundation Conference*. 2001.

Hudson, Graham, Alain Leger, Birger Niss, and Istvan Sebestyen. "JPEG At 25: Still Going Strong." *IEEE MultiMedia* 24, no. 2 (2017): 96–103.

Hunt, Andrew, and David Thomas. *The Pragmatic Programmer*. Reading, MA: Addison-Wesley, 1999.

IBM. "Carbon Design System · GitHub." *GitHub*. Accessed January 20, 2020. https://github.com/carbon-design-system.

Interaction Design Foundation. "Should We Introduce Benchmarking for Our UX Research?" *The Interaction Design Foundation*. Accessed December 10, 2019. https://www.interaction-design.org/literature/article/should-we-introduce-benchmarking-for-our-ux-research.

Jobs, Steve. "Thoughts on Flash." Published April, 2010. https://www.apple.com/hotnews/thoughts-on-flash/.

Kexin, Hu. "Advantages and Disadvantages of Modularity." eds. Gunnar Erixon and Patrik Kenger. *2nd Seminar on Development of Modular Products*. Dalarna University, 2004.

Kholmatova, Alla. *Design Systems: A practical Guide to Creating Design Languages for Digital Products*. Freiburg, Germany: Smashing Media AG, 2017.

Kumar, Ashok, Jiju Antony, and Tej Dhakar. "Integrating Quality Function Deployment and Benchmarking to Achieve Greater Profitability." *Benchmarking: An International Journal* 13, no. 3 (2006): 290–310.

Langefors, Börje. *Essays on Infology: Summing Up and Planning for the Future*. ed. Bo Dahlbom. Lund, Sweden: Studentlitteratur, 1995.

Lexico. "Module." Accessed November 23, 2019. https://www.lexico.com/en/definition/module.

Lidwell, William, Kritina Holden, and Jill Butler. *Universal Principles of Design, Revised and Updated: 125 Ways to Enhance Usability, Influence Perception, Increase Appeal, Make Better Design Decisions, and Teach through Design*. Beverly, MA: Rockport Publishers, 2010.

Lunden, Ingrid. "Music streaming pioneer SoundCloud raises $75M from Pandora owner SiriusXM." *TechCrunch*. Published February 11, 2020. https://techcrunch.com/2020/02/11/music-streaming-pioneer-soundcloud-raises-75m-from-pandora-owner-siriusxm/.

Malamed, Connie. *Visual Language for Designers: Principles for Creating Graphics that People Understand*. Beverly, MA: Rockport Publishers, 2009.

McClure, Robert. "Software Engineering Conference." Arizona, 2001.

Madsen, Rune. "What Is a Design System? – Programming Design Systems." Published January 17, 2018. https://programmingdesignsystems.com/what-is-a-design-system/index.html#what-is-a-design-system-OQAaTNQ.

Marshall, Russell, Paul Leaney, and P. Botterell. "Enhanced Product Realisation Through Modular Design: An Example of Product/process Integration." *Third Biennial World Conference on Integrated Design and Process Sciences*. Berlin, Germany, 1998.

Merriam-Webster. "Definition of Module." Accessed August 6, 2019. https://www.merriam-webster.com/dictionary/module.

Miller, James Grier. *Living Systems*. McGraw-Hill, Inc., 1978.

Morreale, Patricia, and Kornel Terplan. *The CRC Handbook of Modern Telecommunications*. 2nd ed. Boca Raton, FL: CRC Press, 2010.

Myers, Michael. *Qualitative Research in Business and Management*. 3rd ed. SAGE, 2020.

Kentaro, Nobeoka, and Michael Cusumano. "Multiproject Strategy and Sales Growth: the Benefits of Rapid Design Transfer in New Product Development." *Strategic Management Journal* 18, no. 3 (1997): 169–186.

Kim, Sung-Eon, Thomas Shaw, and Helmut Schneider. "Web Site Design Benchmarking Within Industry Groups." *Internet Research* 13, no. 1 (2003): 17–26.

Kuznetsov, Gleb. "Visual Design Language: The Building Blocks Of Design." *Smashing Magazine*. Published March 30, 2020. https://www.smashingmagazine.com/2020/03/visual-design-language-building-blocks/#comments-visual-design-language-building-blocks

Online Etymology Dictionary. "Module." Accessed August 6, 2019. https://www.etymonline.com/word/module.

O'Reilly, Kelley, David Paper, and Sherry Marx. "Demystifying Grounded Theory for Business Research." *Organizational Research Methods* 15, no. 2 (2012): 247–62.

Ou, Owen, and Brian H. Kheiner. "Excellence in Benchmarking." *Industrial Engineer* 47, no. 12 (2015): 20–24.

Parnas, David. "On the Criteria To Be Used in Decomposing Systems into Modules." *Communications of the ACM* 15, no. 12 (1972): 1053–1058.

Pate, Alex. "A collection of awesome design systems." *GitHub*. Published June 16, 2017. https://github.com/alexpate/awesome-design-systems.

Patel, Khushboo (Lara), and Brian Kleiner. "The Future of Benchmarking Is Digital." *Industrial Management* 59, no. 4 (2017): 26–30.

Ries, Eric. *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. New York, NY: Crown Business, 2011.

Rodgers, Paul, and Alex Milton. *Product Design*. Laurence King Publishing, 2011.

Rosenbaum, Réne, and Christian Tominski. "Pixels vs. Vectors: Presentation of Large Images on Mobile Devices." *International Workshop on Mobile Computing (IMC)*. 2003.

Rozat, Pascal. "Deezer: Profitability Down the Line?" *Institut National de l'audiovisuel Websites*. Published August 18, 2011; last modified August 19, 2011. https://web.archive.org/web/20150403164334/http://www.inaglobal.fr/en/music/article/deezer-profitability-down-line?tq=4.

Rutherford, Zack. "Design Systems vs. Pattern Libraries vs. Style Guides – What's the Difference?" *UXPin*. Published September 29, 2017; last modified July 9, 2019. https://www.uxpin.com/studio/blog/design-systems-vs-pattern-libraries-vs-style-guides-whats-difference/.

Sanchez, Ron, and Robert Collins. "Competing – and Learning – in Modular Markets." *Long Range Planning* 34, no. 6 (2001): 645–667.

Sanchez, Ron. "Creating Modular Platforms for Strategic Flexibility." *Design Management Review* 15, no. 1 (2004): 58–67.

Schepers, Doug, interview by Jen Simmons. *SVG with Doug Schepers*. Published April 17, 2014.

Schilling, Melissa. "Toward A General Modular Systems Theory And Its Application To Interfirm Product Modularity." *Academy of Management Review* 25, no. 2 (2000): 312–334.

Sheppard, Benedict, Garen Kouyoumjian, Hugo Sarrazin, and Fabricio Dore. "The Business Value Of Design." *McKinsey & Company*. Published October, 2018. https://www.mckinsey.com/business-functions/mckinsey-design/our-insights/the-business-value-of-design.

Siegler, MG. "Microsoft Has Seen The Light. And It's Not Silverlight." Published October 30, 2010. https://techcrunch.com/2010/10/30/rip-silverlight-on-the-web/.

Simon, Herbert. "The Architecture of Complexity." *Proceedings of the American Philosophical Society* 106, no. 6 (1962): 467–482.

Sisario, Ben. "Jay Z Buys the Music Streaming Firm, Aspiro." *The New York Times*. Published March 13, 2015. https://www.nytimes.com/2015/03/14/business/media/jay-z-buys-the-music-streaming-firm-aspiro.html.

Stol, Klaas-Jan, Paul Ralph, and Brian Fitzgerald. *Grounded Theory in Software Engineering Research: A Critical Review and Guidelines*. 2016.

Suarez, Marco, Jina Anne, Katie Sylor-Miller, Diana Mounter and Roy Stanfield. *Design Systems Handbook*. DesignBetter by InVision, 2017.

Ulrich, Karl. "The Role of Product Architecture in the Manufacturing Firm." *Research Policy* 24, no. 3 (1995): 419–440.

Ulrich, Karl. "Design Is Everything?" *Journal of Product Innovation Management* 28, no 3. (2011): 394–398.

Ulrich, Karl, and Steven Eppinger. *Product Design and Development*. 5th ed. New York, NY: McGraw-Hill, 2012.

Waterman, Michael, James Noble, and George Allan. *How Much Up-Front? A Grounded Theory of Agile Architecture*. 2015.

Watson, Gregory H. *Strategic Benchmarking Reloaded with Six Sigma: Improving Your Company's Performance Using Global Best Practice*. John Wiley & Sons, 2008.

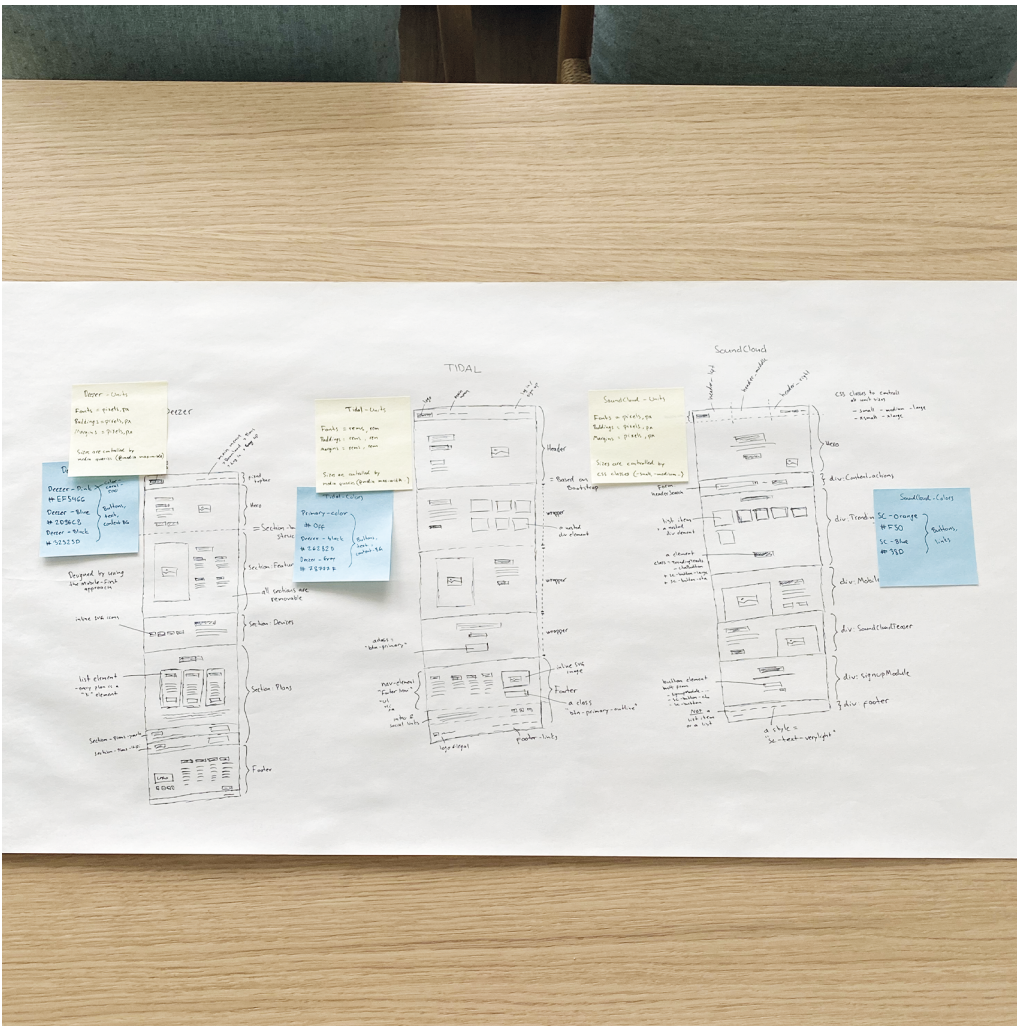
World Wide Web Consortium (W3C). "Facts About 3WC." Accessed April 26, 2019. www.w3.org/Consortium/facts.

W3Schools. "CSS Syntax and Selectors." Accessed August 5, 2019. https://www.w3schools.com/css/css_syntax.asp.

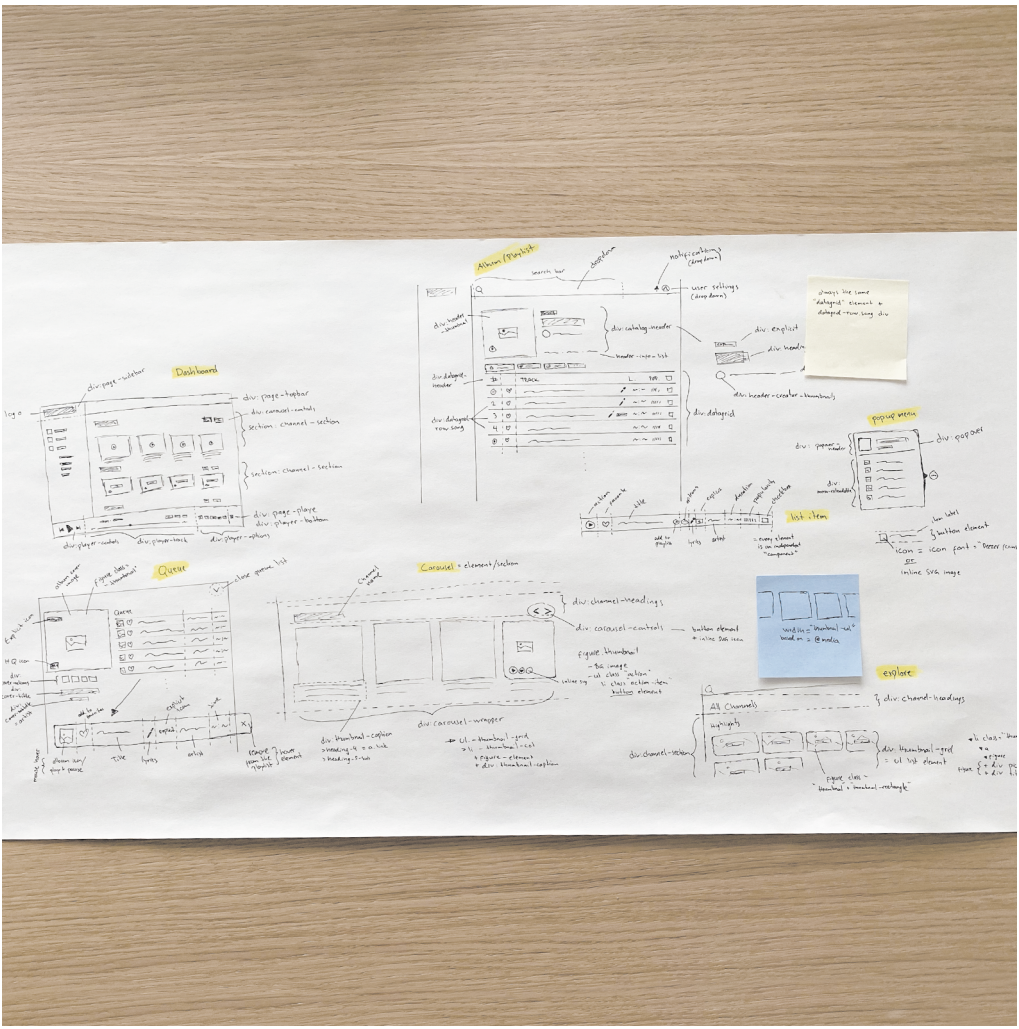
Yoo, Noah. "The Full Transcript Of Jay Z's Tidal Q&A At The Clive Davis Institute Of Recorded Music." *The Fader*. Published May 1, 2015. http://www.thefader.com/2015/04/01/the-full-transcript-of-jay-zs-qa-at-the-clive-davis-institute-of-recorded-music.

Appendix: Behind the Scenes

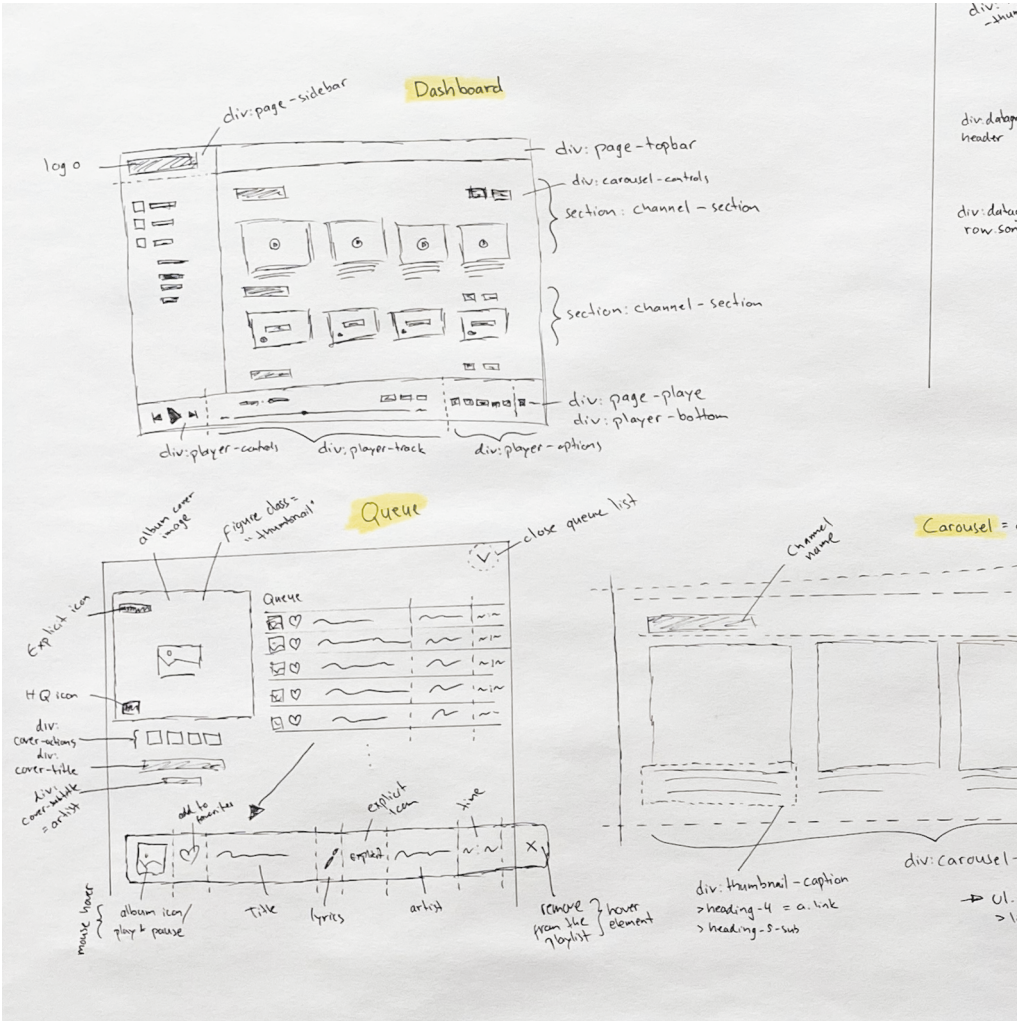
All appendices related
to the research part of this thesis.



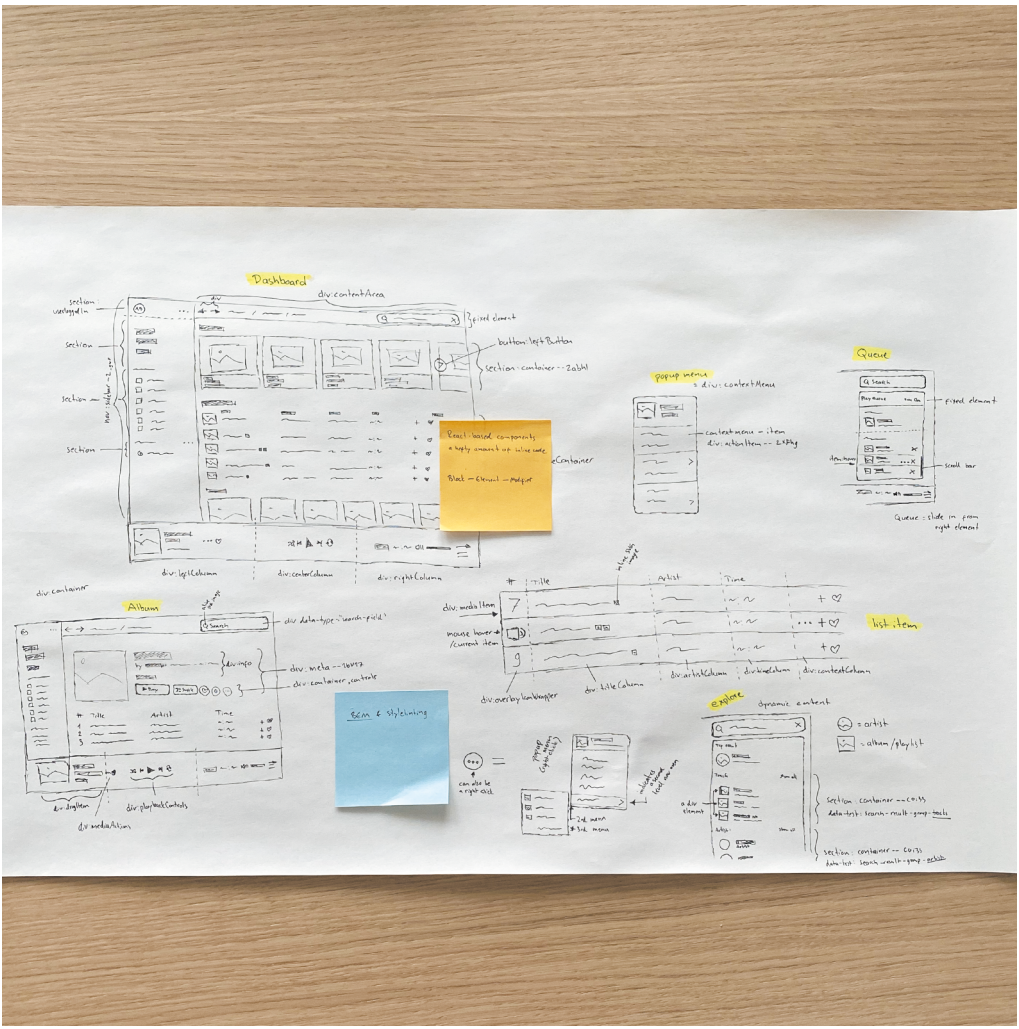
Wireframes of the selected streaming services Deezer, Tidal, and SoundCloud.



Deezer's wireframes.

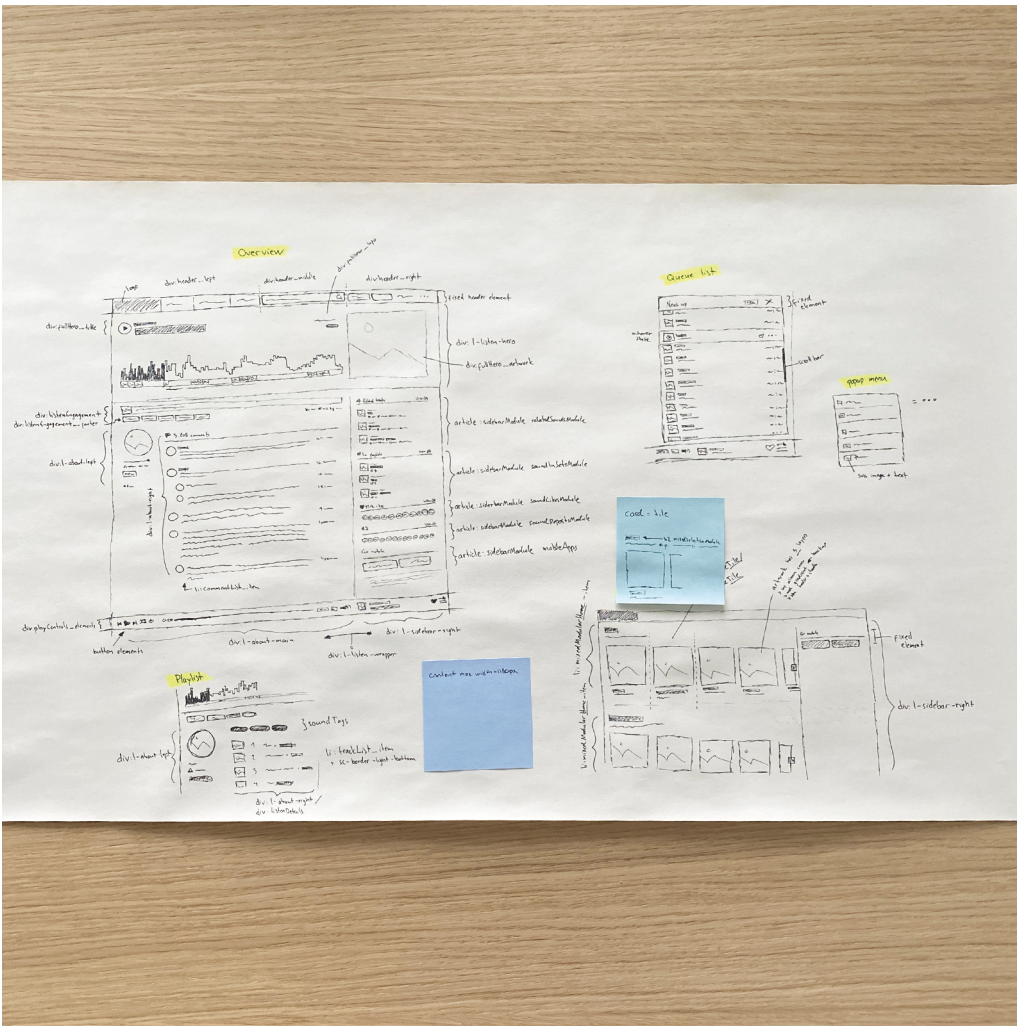


A close-up of Deezer's wireframes.

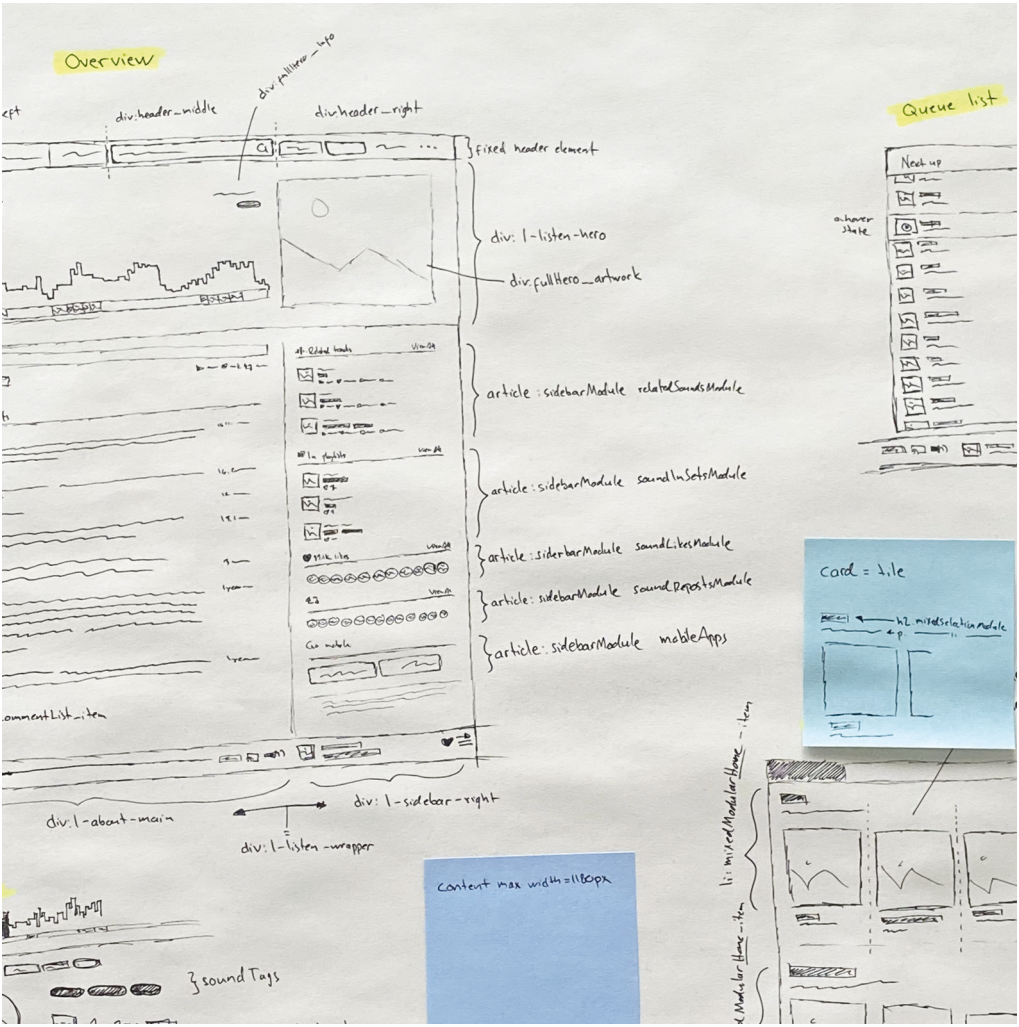


Tidal's wireframes.

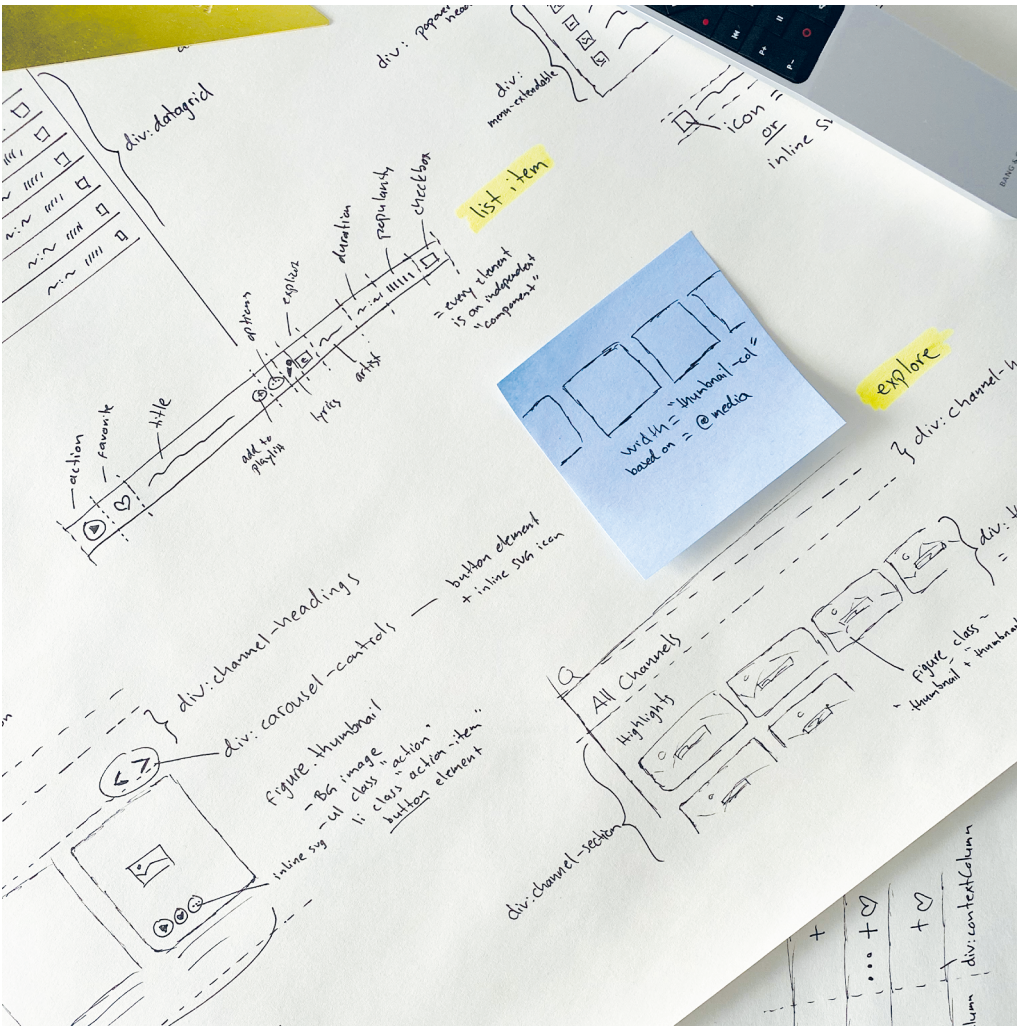
A close-up of Tidal's wireframes.



SoundCloud's wireframes.



A close-up of SoundCloud's wireframes.



The core focus was on structural elements and definitions; all wireframes examined how the selected streaming services were designed and developed.

A collage of notes, source codes, and layouts that guided the research process. The materials were mainly created during the coding phase.

2020 / Christian Anttonen